MARINE CORPS TACTICAL SYSTEMS SUPPORT ACTIVITY CAMP --ETC F/G 9/2
DIGITAL COMMUNICATIONS TERMINAL HIGH ORDER PROGRAMMING LANGUAGE--ETC(11)
NOV 80 K C SHUMATE, R E SAUER, G E ANDERSON
24E005/U-TN-01-VOL-2 AD-A118 811 UNCLASSIFIED NL 1 o **2** Tiesn

UNCLASSIFIED



REPORT DOCUMENTATION P		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER 2.	GOVT ACCESSION NO.	2. RECIPIENT'S CATALOG NUMBER
TECHNICAL NOTE 24E005/U-TN-01	D-A118811	
4. TITLE (and Subtitle)		S. TYPE OF REPORT & PERIOD COVERED
DIGITAL COMMUNICATIONS TERMINAL HIGH PROGRAMMING LANGUAGE STUDY (DCT HOL:		Final report, Vol. 2
VOLUME TWO: APPENDICES	·	6. PERFORMING ORG. REPORT NUMBER
, · · · ·	. HOOPER USMC F. BRADY USMC	8. CONTRACT OR GRANT NUMBER(4)
5. PERFORMING ORGANIZATION NAME AND ADDRESS		16. PROGRAM ELÉMENT, PROJECT, TASK AREA & WORK UNIT HUMBERS
Marine Corps Tactical Systems Suppor Marine Corps Base Camp Pendleton, CA 92055	t Activity	DCT Project
11. CONTROLLING OFFICE NAME AND ADDRESS		12. REPORT DATE
Marine Corps Development and Educati	on Command	26 November 1980
Quantico, VA 22134		13. Number of Pages 100
14. MONITORING AGENCY HAME & ADDRESS(If different to	rem Controlling Office)	18. SECURITY CLASS. (of this report)
Marine Corps Tactical Systems Suppor Marine Corps Base	t Activity	UNCLASSIFIED
Camp Pendleton, CA 92055	:	18a. DECLASSIFICATION/DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of Mis Report)		

Approved for Public Release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, If different from Reg

Approved for Public Release; distribution unlimited.

IS. SUPPLEMENTARY NOTES

See also Volume 1.

Copy available to DTIC does not permit fully legible reproduction

HIGH LEVEL LANGUAGE

DIGITAL COMMUNICATIONS TERMINAL (DCT)

HIGH ORDER PROGRAMMING LANGUAGE

NSC800 MICROPROCESSOR

PROGRAMMING LANGUAGE

COMPUTER PROGRAMS

DCT HOL STUDY **MICROPROCESSORS**

DECISION MAKING

DELPHI

This paper reports the results of a study to select a high order programming language for the development of computer programs for the digital communications terminal. All languages suitable for use with the NSC800 microprocessor were considered. The nine final candidates were evaluated by a methodology including benchmarking and determination of a figure of merit. During the conduct of the study it became clear that the program support environment must include both a minicomputer software engineering host, and a microcomputer development

DD , FORM 1473

EDITION OF ! NOV 65 IS OBSOLETE 5/N 8102-UF-014-4401

UNCLASSIFIED

SECURITY CLASSFICATION OF

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (Then Date Enter

20. ABSTRACT

system. The language selected is Interactive Systems C. The system includes a cross-compiler running on a PDP-11 and generating code for the NSC800.

S/N 0102- LF- 014- 4601

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PACE(When Date Shiered

DISCLAIMER NOTICE

THIS DOCUMENT IS BEST QUALITY PRACTICABLE. THE COPY FURNISHED TO DTIC CONTAINED A SIGNIFICANT NUMBER OF PAGES WHICH DO NOT REPRODUCE LEGIBLY.

TECHNICAL NOTE 24E005/U-TN-01

DIGITAL COMMUNICATIONS TERMINAL HIGH ORDER PROGRAMMING LANGUAGE STUDY

(DCT HOL STUDY)
VOLUME TWO: APPENDICES
26 NOVEMBER 1980



MARINE

C

0

CORPS

TACTICAL

SYSTEMS



SUPPORT

ACTIVITY

CAMP PENDLETON

CALIFORNIA 92055

Approved for public release; distribution unlimited.

82 08 31 022

TABLE OF CONTENTS

			Page
	APPENDIX A	REFERENCES	A-1
	Tab 1 Tab 2 'Tab 3	Primary References Periodicals General Bibliography	A-4
i o	Tab 4	Language Documentation	
	APPENDIX B	BENCHMARK PROGRAMS	B-1
©	Tab 1 Tab 2 Tab 3 Tab 4	Program DCT BENCHMARK (Commented) Program DCT BENCHMARK (Code Only) Error-Seeded DCT BENCHMARK Interactive "C" Source	B-9 B-12 B-16
•	Tab 5 Tab 6 Tab 7 Tab 8	Whitesmith "C" Source	B-24 B-28 B-33
; 	Tab 9 Tab 10 Tab 11 Tab 12 Tab 13	PASCAL/Z Source PLI-80 Source (Version 1) PLI-80 Source (Version 2) PLMX Source PLZ Source	B-41 B-46 B-50
°()	APPENDIX C	BENCHMARK RESULTS	C-1
	APPENDIX D	DELPHI PHASE 1: WEIGHTS	D-1
	APPENDIX E	DELPHI PHASE 2: SCORES AND FIGURES OF MERIT	E-1
C		LIST OF ILLUSTRATIONS	
	Figure	Title	Page
. 0	C-1	Benchmark Results	C-3
O	D-1	Ranking of Features	D-3
•	E-1	Figures of Merit without Benchmark Values	E-6
• -	E- 2	Final Figures of Merit	E-7

0 .

LIST OF TABLES

$\mathbb{Q}_{\mathbb{Q}}$	Table	Title	Page
	C-1	DCT HOL Summary Benchmark Results	C-2
	D-1	Phase 1 Delphi Study Statistical Summary	D-2
Э	D-2	First Iteration DCT HOL Language Feature Response Summary	D-4
	D-3	Second Iteration DCT HOL Language Feature Response Summary	D - 5
9	D-4	Third Iteration DCT HOL Language Feature Response Summary	D-6
	E-1	Delphi Study Phase 2 Summary (1st Iteration)	E-2
	E-2	Delphi Study Phase 2 Summary (2nd Iteration)	E-3
3 .	E-3	Delphi Study Phase 2 Summary (3rd Iteration MCTSSA)	E-4
	in .	Delphi Study Phase 2 Summary (3rd Iteration NOSC Included)	E-5

Acces	sion For		
NTIS	GRA&I	M	
DTIC	TAB	1	
Unann	ounced		1
Justi	fication_		_
	ibution/ lability	Codes	DTIG
	Avail and	l/or	INSPECTE
Dist	Special	L	2
Ω	23		

Ö

....

APPENDIX A BIBLIOGRAPHIC INFORMATION

This appendix contains four tabs. Tab 1 presents the documents and articles which were of particular value in the preparation of the report. Tab 2 lists periodicals which were reviewed for advertising and general market information regarding available HOLs. Tab 3 is a detailed bibliography of the trade journal literature on high order languages. Tab 4 is a list of the publications obtained from language vendors which are references for the languages which were evaluated.

Tab 1

Primary References

(a) Cherlin, Mokurai, "High-Level Languages for Microcomputers,"

Mini-Micro Systems, April 1980, pp. 92-103.

C

0

. 3

0

0

O

- (b) Considerations for a Microprocessor Policy, with appendixes, MCTSSA, 18 June 1980.
- (c) Harvany, Jozsef and Jozsef Janos, "Software Products for Manufacturing, Design, and Control," <u>Proceedings of the IEEE</u>, Vol. 68, No. 9, September 1980, pp. 1050-1053.
- (d) Meyers, Edith, "Software Slump in Sight?" <u>Datamation</u>, May 1980, pp. 93-94.
- (e) "More and Better Operating Systems Turn More Chips into Better μCs," <u>Electronics</u>, Vol. 28, No. 6, 15 March 1980, pp. 223-224.
- (f) Nichols, J. E., The Structure and Design of Programming Languages,
 Addison-Wesley, 1975, p. 8.
- (g) Pratt, T. W., <u>Programming Languages: Design and Implementation</u>, Prentice-Hall, 1975, p. 207.
- (h) Reghiszi, Stefano Crespi, "A Survey of Microprocessor Languages,"

 Computer, Jan 1980, p. 48.
- (i) Sabin, M.A., "Portability -- Some Experiences with FORTRAN,"

 <u>Software Practice Experience</u>, Vol. 6, July September 1976, pp. 393-396.
- (j) Schindler, Max, "Pick a Computer Language That Fits the Job," Electronics, 19 July 1980, pp. 62-78.

(k) Schindler, Max, "UNIX with Workbench Will Serve Superminis," Electronics, 8 November 1979, pp. 30-31.

0,

9

. 0

0

0

O

(1) Stiefel, Malcom L., "A Guide to Tool Selection," <u>Mini-Micro</u>

<u>Systems</u>, August 1980, pp. 68-76.

Tab 2 Periodicals

- 1. Computer Design: The Magazine of Computer-Based Systems, Vol 19, No. 5, May 1980.
- 2. Computer Design: The Magazine of Computer-Based Systems, Vol 19, No. 6, June 1980.

0

: 0

0

0

ر.

0

. 🔿

- 3. Computer Design: The Magazine of Computer-Based Systems, Vol 19, No. 7, July 1980.
- 4. Electronic Design, Vol. 28, No. 6, March 15, 1980.
- 5. Electronic Design, Vol. 28, No. 12, June 7, 1980.
- 6. Electronic Design, Vol. 28, No. 13, June 21, 1980.
- 7. Electronic Design, Vol. 28, No. 14, July 5, 1980.
- 8. Electronic Design, Vol. 28, No. 16, August 2 1980.
- 9. Electronics: The International Magazine of Electronic

 Technology, Vol. 53, No. 9, April 17, 1980.
- 10. Electronics: The International Magazine of Electronic
 Technology, Vol. 53, No. 12, May 22, 1980.
- 11. Electronics: The International Magazine of Electronic
 Technology, Vol. 53, No. 13, June 5, 1980.
- 12. Electronics: The International Magazine of Electronic
 Technology, Vol. 53, No. 16, July 17, 1980.
- 13. Electronics: The International Magazine of Electronic

 Technology, Vol. 53, No. 17, July 31, 1980.

14. Mini-Micro Systems, Vol. XIII, No. 4, April 1980.

0 ,

0

. 🕝

ು

0()

- 15. Mini-Micro Systems, Vol. XIII, No. 6, June 1980.
- 16. Mini-Micro Systems, Vol. XIII, No. 7, July 1980.

Tab 3

General Bibliography

The information to prepare this bibliography of high order languages was obtained by researching the journals listed below, representing the period March through September 1980:

Byte
Communications of the ACM
Computer Design
Datamation
EDN
Electronics
Electronics Design
Mini-Micro Systems

0

0

 \odot

- High Order Languages -- General Information: Journal articles that contain surveys, comparisons, discussion and general information about the various high order Languages.
- High Order Languages -- Specific: Journal articles that contain information about specific language characteristics and their use in applications programming.
- Microcomputer Development Systems: Journal articles that contain information about the hardware and software characteristics of the various state-of-the-art MDSs that are currently available.
- In Circuit Emulation: Journal articles that discuss the uses and types of in-circuit emulator as a product-development and interactive debugging facility.
- High Order Language Philosophy: A collection of material by several authors discussing the relative merits of using HOLs for applications development.

High Order Languages; General Information

- 1. Cherlin, Modurai, "High-level Languages for Microcomputers," Mini-Micro Systems, Apr 1980, p.89.
- 2. Feldman, Jerome A., "High Level Programming for Distributed Computing," Communications of the ACM, Vol 22, Jun 1979, p.353.
- 3. Hughes, Philip, "Macro Assembers for Micros," Mini-Micro Systems, Apr 1979, p.71.
- 4. Kroeker, Edwin J., "Multilingual Software Cuts Development Costs," Mini-Micro Systems, May 1980, p.163.
- 5. Morris, Robert A., "Comparison of Some High-Level Languages," Byte, Feb 1980, p.128.
- 6. Ogdin, Carol Ann, "The Many Choices In Development Languages," Mini-Micro Systems, Aug 1980, p.81.
- 7. Reghizzi, Stefano Crespi, "A Survey of Microprocessor Languages," Computer, Jan 1980, p.48.
- 8. Schindler, Max, "Check Operating-System Features To See How Powerful a uC Will Be," <u>Electronic Design</u>, Nov 22, 1979, p.133.
- 9. Schindler, Max, "Pick A Computer Language That Fits The Job," Electronic Design, Jul 19, 1980, p.62.
- 10. Wecker, D.B., R.L., Krutz, & D.T., Tuma, "High Level Design Language Develop Low Level Microprocessor - Independent Software," Computer Design, Jun 1979, p.140.

Product Advertisement

0

0

O

 \circ ()

 \circ

3

C.

11. Scientific Enterprises, Inc., "The Software Synthesizer; Lets the Engineer Take a Hardware Approach to Program Development,"

Mini-Micro Systems, Jun 1980, p.45.

High Order Languages; Specific

ADA

1. Glass, Bob, "From PASCAL to Pebbleman and Beyond," <u>Datamation</u>, Jul 1979, p.146.

APL

¹0

()

 \odot

<u>'</u>

2. Gull, W.E. & M.A., Jenkins, "Recursive Data STructures in APL," Communications of the ACM, Vol 22, Jan 1979, p.79.

FORTH

- 3. Hicks, Stephen M., "FORTH's Forte is Tighter Programming," Electronics, Mar 15, 1979, p.114.
- 4. Mannoni, Michel, "Forth An Extensible Path to Efficient Programs," <u>Electronic Design</u>, Jul 19, 1980, p.175.
- 5. James, John S., "What is FORTH? A Tutorial Introduction," Byte, Aug 1980, p.100.
- 6. Harris, Kim, "FORTH Extensibility, or how to Write a Compiler in 25 Words or Less," Byte, Aug 1980, p.164.
- 7. Miller, Richard & Jill Miller, "Breakforth into FORTH!", Byte, Aug 1980, p.150.
- 8. Moore, Charles H., "The Evolution of FORTH, an Unusual Language,"
 Byte, Aug 1980, p.76.
- 9. Williams, Gregg, "FORTH Glossary," Byte, Aug 1980, p.186.
- MDL Modular Development Language
- 10. Schnabel, Dennis, "MDL/u A New Language for Effective uP Software Development," <u>Electronic Design</u>, Sep 13, 1979, p.102.
- MPL Developed by Motorola
- 11. Waller, Larry, "Microcomputers to Run Test Hours," Electronics, Jul 19, 1979, p.92.

PASCAL

- 12. Conrad, Marvin, "PASCAL A High Level Language for Micros & Minis," <u>Datamation</u>, Jul 1979, p.153.
- 13. Doty, Keith L., "A Top Down Evaluation of Pascal," Computer Design, May 1980, p.167.

- 14. Fletcher, Dennis, "PASCAL Power," Datamation, Jul 1979, p.142.
- Hemenway, Jack & Edward Teja, "PASCAL Update," EDN, Apr 5, 1980, p.100.
- 16. Mundie, David A., "PASCAL and the Great Race," Byte, Sep 1980, p.94.
- 17. Shillington, Keith, "Structure: The Key to Pascal's Problem-Solving Power," <u>Datamation</u>, Jul 1979, p.151.

PL/1

 \circ

)

00

0

0

0

18. Marshall, Martin, "PL/1 Seeks Popularity," Electronics, Apr 24, 1980, p.102.

PLZ

19. Riggins, Chris, "When No Single Language Can Do the Job, Make it a Language-Family Matter," <u>Electronic Design</u>, Feb 15, 1979, p.86.

PILOT/P

20. Mundie, David, "PILOT/P: Implementing a High-Level Language in a Hurry," Byte, Jul 1980, p.154.

ROSETTA

21. Warren, Scott K. & Abbe, Dennis, "Presenting ROSETTA Smalltalk,"

<u>Datamation</u>, May 1980, p.145.

TINY "C"

22. Kern, Christopher O, "A User's Look at Tiny - C," Byte, Dec 1979, p.196.

Product Advertisements

- 23. Analog Devices Inc., "A PL uS (A Programming Language for Microprocessor Systems)," <u>Electronics</u>, Nov 8, 1979, p.41.
- 24. Vanguard Systems Corp, "APL/DTC Desktop Microcomputer using APL,"

 Computer Design, Aug 1979, p.190.
- 25. Whitesmiths', "Whitesmiths' full C Compiler," Electronics, Sep 13, 1979, p.199.
- 26. Micro Focus, Ltd., "COBOL for 8 & 16 Bit Microcomputers,"
 Mini-Micro Systems, Mar 1980, p.79.
- 27. Systems Engineering Laboratories Inc., "FORTRAN Compiler (Surpasses the ANSI 77 Standard)," <u>Electronics</u>, Aug 16, 1979, p.155.
- 28. Bally Corp., "GRAFIX, A Self Teaching, User-Expandable Language,"
 Electronics, Jan 18, 1979, p.48.

29. SofTech Microsystems, "PASCAL Confusion Explained," Datamation, May 1980, p.98.

0

1

0

 $\circ()$

0

0

- 30. Control Systems Inc., "PASCAL resides in PROM; UCSD Pascal Firmware for 6800 and 6809 Processors Eliminates Interpreter Software," Electronics, Apr 26, 1979, p.212.
- 31. John Wiley & Sons, Inc. (Book Publishers), "An Introduction to Problem Solving & Programming with PASCAL & PL/1 Structured Programming, 2nd Ed.," Communications of the ACM, Jun 79.
- 32. Zilog, Inc., "Structured Z8O PASCAL for Microcomputers," Computer Design, Sep 1979, p.166.
- 33. Digital Research, "PL/1 Shrinks to fit Microprocessors," Electronics, Apr 24, 1980, p.41.
- 34. Mostek, "PL/M Development Languages," Electronics, Jan 4, 1979, p.33.
- 35. General Health, "SIMPL, A High Level Interactive Microprocessor Language," Electronics, Jan 17, 1980, p.40.

Microcomputer Development Systems

 \bigcirc

 \odot

•

3

 \bigcirc

0

)

- 1. Bailey, Chris & Tracy Kahl, "Evaluation Delay Cuty By Low-Cost Microprocessor Development Tool," Electronics, Aug 30, 1979, p.121.
- 2. Gladstone, Bruce E, "Comparing Microcomputer Development System Capabilities," Computer Design, Feb 1979, p.83.
- Lee, Edwin, "Debunking The Development System Myth (Opinion)," Mini-Micro Systems, Aug 1980, p.107.
- 4. McCracken, David, "Hybrid Tool for Universal Microprocessor Development," Computer Design, Apr 1980, P.199.
- 5. McLeod, Jonah, "uC Development Systems Have The Hardware, Software 16-Bit uP's Need," <u>Electronic Design</u>, Sep 1979, P.92.
- 6. Stiefel, Malcolm L., "A Guide To Tool Selection," Mini-Micro Systems, Aug 1980, p.68.
- 7. Stiefel, Malcolm L., "Microcomputer Development Systems (Survey),"
 Mini-Micro Systems, Sep 1979, p.74.
- 8. Stock, Bruce, "A Centralized Design Support Center," Mini-Micro Systems, Aug 1980, p.87.

Product Advertisements

- 9. Microsoft, "MS-Pascal Compiler, Optimiser & Modular Code Generators for 8080, Z80, 8086, A8000, and a Pseudo Machine," Computer Design, Aug 1980, p.152.
- 10. Phoenix Digital Corp, "EXOR 80/XASM Assembles 280 and 8080 Source Programs into Machine Code," Computer Design, Dec 1979, p.132.
- 11. Infosoft Systems, Inc. "Microcomputer OS for 8080, 8085, and Z80 based Microcomputers, the I/OS Disk Operating System," <u>Datamation</u>, Unk, p.260.
- 12. Pascal Development Company, "PASCAL & 8002 Combine in Microprocessor Development," <u>Electronics</u>, Jul 5, 1979, p.206.
- 13. Danfysik A/S, "The 'UNIQUE' Microprocessor Development System," Electronics, Jul 5, 1979, p.76.
- 14. Boston Systems Office, "Universal Microprocessor Development System (UMDS)," <u>Electronics</u>, Aug 30 1979, p.226.

In Circuit Emulation

0

0

0

•

0

0

0

- 1. Kelly, James M., "Cut Hardware, Software Development Costs-Take Advantage of In-Circuit-Emulators," <u>Electronic Design</u>, Aug 16, 1979, p.66.
- 2. Moon, Jim, "Microcomputer For Emulation Bares Hidden Buses, Functions," <u>Electronics</u>, Jul 17, 1980, p.126.
- Saponas, Tom, "Development System: Soft Keys + Four Buses = Easy Use and Full-Speed Emulation," <u>Electronic Design</u>, Sep 27, 1980, p.38.

Product Advertisements

- 4. Applied Microsystems, "EX-80 a Z80 Emulator Frees MDS By Taking On Test Chores," <u>Electronics</u>, Apr 12, 1979, p.240.
- 5. Tecma Inc., "The Microsystem Emulator; A Versatile In-Circuit Emulator Complements Any uP Development System," EDN, Sep 20, 1979, p.71.

High Order Language Philosphy

0

0

- 1. Bai, Subhash, Yoav Lavi, Asher Kaminker, and Avram Menachem, "Optimizing Microprocessor Performance," Mini-Micro Systems, Jun 1980. p.103.
- 2. Brooks, Ruven E., "Studying Programmer Behavior Experimentally: The Problems of Proper Methodology," Communications of the ACM, Vol 23, Apr 1980, p.207.
- 3. Caffin, R.N., "The Open Channel," Computer, Mar 1979, p.108.
- 4. Caudill, Pat, "Using Assembly Coding to Optimize HIgh-Level Language Programs," <u>Electronics</u>, Feb. 1979, p.121.
- 5. Ehram, John R. "The New Tower of Babel," <u>Datamation</u>, Mar 1980, p.157.
- 6. Esterling, Bob, "Software Manpower Costs: A Model," <u>Datamation</u>, Mar 1980, p.164.
- 7. Heckel, Paul, "Developing Software or Microprocessor-Based Products," Mini-Micro Systems, Feb 1980, p.111.
- 8. Hug, Richard & Dr. Presser, Leon, "Designing Transportable Software," Mini-Micro Systems, May 1980, p.122.
- 9. Posa, John G., "Microprocessors and Microcomputers," Electronics, Oct 25, 1979, p.144.
- 10. Posa, John G., "Programming Microcomputer Systems with High-Level Languages," Electronics, Jan 18, 1979.
- 11. Sheppard, Sylvia B., Curtis, Bill, Milliman, Phil, & Love, Tom,
 "Modern Coding Practices and Programmer Performance," Computer, Dec
 1979, p.41.
- 12. Turner, Joshua, "The Structure of Modular Programs," Communications of the ACM, Vol 23, May 1980, p.272.
- 13. Wulf, William A., "Trends in the Design and Implementation of Programming Languages," Computer, Jan 1980, p.14.
- 14. Klatt, Garth, "PL/1 Vs. PASCAL (Letter to the ED)," <u>Datamation</u>, Jan 1980, p.35.
- 15. Amort, Anthony, "APL Enthusiast (Letter to the ED)," <u>Datamation</u>, Mar 1980, p.39.

Miscellaneous Material

Ö

0

2(

0

0

- 1. Allison, Andrew, "Setting Standards for Microprocessors," Mini-Micro Systems, Oct 1979, p.66.
- Dwyer, Ed, Michel Lafortune, & Michel Bertrand, "Bootstrap Lets Multibus Tap CP/M-Based Software," <u>Electronic Design</u>, May 10, 1980, p.219.
- 3. Grappel, Rober & Jack Hemenway, "The MC68000 A 32-bit uP Masquerading as a 16-bit Device," EDN, Feb 20, 1980, p.127.
- 4. Lowe, Linda, "System for Terminals Creates 'Keyboards' Anyone Can Use," Electronics, Jun 5, 1980, p.39.
- 5. Moore, Cecil A., "Ready-made Multiplexer Simplifies Multitasking," Electronic Design, Jul 19, 1980, p.153.

Product Advertisements etc.

- 6. Electro, "Summation of Electro/80 Conference & Exhibition," Electronics, Apr 24, 1980, p.142.
- 7. Western Digital Corporation, "Chip Set and Development Computer Execute PASCAL Object Programs," Computer Design, May 1979, p.250.
- 8. Texas Instruments Inc., "TI To Add PASCAL to its FS990 Development Unit," <u>Electronics</u>, Mar 26, 1979, p.36.
- 9. Advanced Micro Computers, "Z8000 Gets Development Unit, Especially Designed for 16-bit Microprocessor, Unit supports 8-bit Processors Too," Electronics, Mar 29, 1979, p.138.
- 10. Mackintosh Publications Ltd., "Microcomputer Analysis Report indicating the Major Growth in Microprocessor Sales in the next few years will probably be in the area of 16-bit Microprocessors," Electronics, Sep 17, 1979, p.261.
- 11. National Semiconductor Corp., "Announces the NSC800 Chip; Which Is Built Around the Architecture of Intel Corp.'s 8085 and executes the Instruction Set of Zilog Inc.'s Z80," Electronics, Oct 11, 1979, p.46.
- 12. Monolithic Systems Corp., "MSC 8004; A Multibus/Z80A Based Microcomputer that Offers 32-Bit Floating Point Arithmethic," Computer Design, Mar 1979, p.210.

Tab 4 Language Documentation

The following documents have been ordered to support this study effort and have been received. They are listed in alphabetical order according to title.

ACT Assembler User's Manual, Version 1.0, Sorcim, June 1980.

Application Note Index, Hewlett-Packard, nd.

3

0

0()

- 3

0

3

ં

"BC -- An Arbitrary Precision Desk-Calculator Language," Lorinda Cherry and Robert Morris, Bell Laboratories, nd.

BDSC User's Guide Addenda v1.32 Edition, Leor Zolman, BD Software, 1980.

BD Software C Compiler v1.3 User's Guide, Leor Zolman, Lifeboat Associates, 1979.

Beginner's Guide for the UCSD PASCAL System, Kenneth L. Bowles, Byte Books, 1980.

C Compiler Systems Interface Manual for 8080 Users, Whitesmiths, Ltd., 1980.

C Compiler User's Manual, Whitesmiths, Ltd., 1980.

C Compiler v1.3 User's Guide, Zolman, Leor, BD Software, 1979.

"Computer Modules," Product Brief, Zilog, Inc., May 1980.

Context Editor for the CP/M Disk System User's Manual, A. Digital Research, 1980.

The C Programming Language, Brian W. Kerninghan and Dennis M. Ritchie, Prentice-Hall, 1978.

CP/M Assembler User's Guide, Digital Research, 1978.

C

3

Э

0()

0

0

0

0

CP/M Dynamic Debugging Tool (DDT) User's Guide, Digital Research, 1978.

CP/M Features and Facilities, An Introduction to, Digital Research, 1980.

CP/M MAC Macro Assembler Language Manual and Applications Guide, Digital Research, 1980.

CP/M SID Symbolic Instruction Debugger User's Guide, Digital Research, 1978.

CP/M 2.0 Interface Guide, Digital Research, 1979.

CP/M 2.0 User's Guide, Digital Research, 1979.

CP/M 2.2 Alteration Guide, Digital Research, 1979.

CP/M ZSID Symbolic Instruction Debugger, Z-80 Version, Digital Research, 1979.

C Reference Manual, Dennis M. Ritchie, Bell Telephone Laboratories, 1975.

CR Z80: BSO Relocating Cross Assembler (with Cross Reference), The Boston Systems Office, Inc., 24 July 1980.

"DC -- An Interactive Desk Calculator," Bell Laboratories, nd.

EDIT 79: A STOIC II Programming Example, Jeffrey L. Zurkow, Avocet Systems. Inc., 1979.

THE REAL PROPERTY.

8080/8085 Assembly Language Programming, Intel Corp., 1979.

Electronic Instruments and Systems: 1980, Hewlett-Packard, nd.

"FORTH: A Cost Saving Approach to Software Development," Elizabeth D. Rather and Charles H. Moore, FORTH, Inc., 1976.

FORTRAN-77 User's Manual, Softech, 1980.

0

0

0

O.

0

FORTRAN-80 Reference Manual, MicroSoft, 1979.

FORTRAN-80 User's Manual, MicroSoft, 1979.

"FORTRAN-80 8080/8085 ANS FORTRAN-77 Intellec Resident Compiler," Intel Corp., 1978.

FORTRAN IV, Cromemco, Inc., 1979.

FORTRAN Lanuage Manual, Zilog, Inc., September 1979.

FORTRAN User Guide, Revision A, Zilog, Inc., October 1978.

FORTRAN User Reference Manual, Softech, 1980.

Getting Started with PASCAL/64000, Hewlett-Packard, nd.

GNAT System 10 Operator's Handbook, GNAT Computers, 1980.

GNAT System 10 Reference Manual, Version 1.2, GNAT Computers, March 1980.

A Guide to Intellec Microcomputer Development Systems, Daniel D. McCracken, Intel Corp., 1978.

A Guide to PL/M Programming for Microcomputer Applications, Daniel D. McCracken, Addison-Wesley, 1978.

INFOSOFT Assembler Manual, Infosoft, 1979.

0

0

0

 \cdot

00

C

0

C

0

"Interactive display Terminal (IDT) Users' Manual," Litton Data Systems, September 1977.

Intersystems Pascal/Z: A High Level Programming Language, Intersystems, 1980.

Introduction to RIO Text Processing, Zilog, Inc., July 1979.

An Introduction to STOIC, Jonathan M. Sachs, Biomedical Engineering Center for Clinical Instrumentation, March 1978.

Intellec Series II Microcomputer Development System Hardware Reference Manual, Intel Corp., 1980.

Introduction to Microprocessor Components Using PL/Z. Conway et al, Winthrop, 1979.

ISIS-II FORTRAN-80 Compiler Operator's Manual, Intel Corp., 1978.

Link-80 Operator's Guide, Digital Research, April 1980.

"Logic Analyser Model 1615A," Technical Data, Hewlett-Packard, 15 Mar 1980.

"Logic State Analyser for Microprocessor Based Systems: Model 1611A," Technical Data, Hewlett-Packard, 15 December 1979.

"Logic State Analyzer for Mini/Micro Computer and Random Logic Analysis: Models 1610A/B," Technical Data, Hewlett-Packard, 15 June 1979.

"A Manual for the Tag Compiler-Writing Language," M.D. McIlroy, Bell Laboratories, 13 September 1972.

"MCZ-1/50: Z80 Microcomputer Systems," Product Brief, Zilog, Inc., November 1979.

"MCZ-1/70: Z80 Microcomputer System," Product Brief, Zilog, Inc., November 1979.

Microcomputer Components Data Book, Zilog, Inc., February 1980.

Microsoft Utility Software, MicroSoft, 1978.

9

0

()

"The MG Macro Processor," Andrew D. Hall, Bell Telephone Laboratories, Inc., June 1971.

"NROFF Users' Manual," Joseph F. Ossanna, Bell Laboratories, 11 September 1974.

PASCAL/M User's Reference Manual, Sorcim, 1979.

PASCAL/MT User's Manual, MT Microsystems, 1980.

PASCAL User Guide, Zilog, November 1979.

PASCAL User Manual and Report, Kathleen Jensen and Niklaus Wirth, Springer-Verlag, 1974

PASCAL/Z User's Manual, Jeff Moskow, 1980 (also titled: Intersystem's PASCAL/Z, A High Level Programming Language, Version 3.0).

"PLM-80 High Level Programming Language Intellec Resident Compiler," Intel Corp., September 1976.

PL/1-80 Applications Guide, Digital Research, 1980.

PL/1-80 Applications Manual, Digital Research, 1980.

PLMX User's Guide, Systems Consultatns, Inc., 1980.

PL/M-80 Programming Manual, Intel Corp., 1978.

PLZ Linker User Guide, Revision A, Zilog, Inc., March 1980.

PLZ Version 3 User's Guide, Zilog, Inc., 1979.

"The Portable C Library (on UNIX)," M. E. Lesk, Bell Laboratories, nd.

Programming in C-A Tutorial, Brian W. Kernighan, Bell Laboratories, 1975.

"PROM User's Manual, Revision A, Zilog, July 1978.

RATFOR-A Preprocessor for a Rational FORTRAN, Brian W. Kernighan, Bell Laboratories, 1975.

RATFOR-80: A High-Level Preprocessor for FORTRAN, Version 1.06, The Software Works, 1 February 1980.

RATFOR Reference Manual, Cromemco, 1979.

7

زر

0

C.

Report on the Programming Language PLZ/SYS, Tod Snook, et al, Springer-Verlag, 1978.

RIO File Debugger: Reference Manual, Zilog, June 1979.

RIO Symbolic Debugger Reference Manual, Revision B, Zilog, November 1978.

Series/80 Board Level Computer Starplex Development System Data Book, January 1980.

Software Tools, Briar W. Kernighan and P. J. Plauger, Addison-Wesley Publishing Co., 1976.

"A System for Typesetting Mathematics," B. W. Kernighan and L. L. Cherry, Bell Laboratories, nd. Systems Data Catalog 1980, Intel Corp., August 1979.

"Tiny PASCAL," SuperSoft, nd.

TSA Rlasm Relocating Linking Macro Assembler and TSA Linka Linking Loader, Tsa Software, Inc., 1979.

Tutorial Introduction to the Unix Text Editor, A, B. W. Kernighan, Bell Laboratories, 1975.

"Typesetting Mathematics -- User's Guide," B. W. Kernighan and L. L. Cherry, Bell Laboratories, nd.

UCSD PASCAL User's Manual, SofTech Microsystems, February 1980.

"UNIX Assembler Reference Manual," Dennis M. Ritchie, Bell Laboratories. nd.

Unix Programmer's Manual, K. Thompson and D. M. Ritchie, Bell Laboratories, 1975.

Using FORTH, Second, Revised Edition, FORTH, Inc., March 1980.

Word Star User's Guide, MicroPro International Corporation, 1980.

"YACC--Yet Another Compiler-Compiler," Stephen C. Johnson, Bell Laboratories, nd.

"ZDS-1 Series," Zilog, Inc., 1977.

 \bigcirc

0

0

Z80 -- Assembly Language Programming Manual, Zilog, Inc, April 1980.

"Z-80 Microcomputer Board Series," Zilog, Inc., 1979.

Z80 Microcomputer Software Programming Guide: Z80 Programming Manual, Mostek Corp., December 1977.

Z80 PLZ Debugging Tool (PDT) User Guide, Zilog, Inc., August 1979.

"280-RIO," Product Specification, Zilog, Inc., July 1979.

Z80-RIO Operating System User's Manual, Revision A, Zilog, Inc., September 1978.

Z80-RIO Text Editor User's Manual, Revision B, Zilog, Inc., 1978.

Z80-RIO Relocating Assembler and Linker User's Manual, Zilog, Inc., 1978.

Z8000 CPU Technical Manual, Zilog, Inc., 1980.

つ

 \supset

C.

"Z8000 Cross-Software Package," Product Brief, Zilog, Inc., May 1980.

"Z8000 Development Module," Product Brief, Zilog, Inc., December 1979.

Product brochures have been received from Dynabyte, Tektronix, Whitesmith Ltd., GNAT computers, Zilog, Avocet Corp., Intel Corp., FORTH, Inc., Heurikon Corp., Infosoft, Hewlett-Packard, Cromemco, MicroSoft, Ontel, Emulogic, Hughes, SuperSoft, and Lifeboat Associates.

APPENDIX B

DCT/HOL STUDY BENCHMARK PROGRAM LISTINGS

This appendix consists of the DCT high order language benchmark program listings for the surviving candidate languages. The listings are divided into two sections. Tabs 1 through 3 contain the pseudo-code specification for the benchmark program. There are three versions of the specification: a code-only version, a heavily commented version, and an error-seeded version.

)

٤)

0()

0

0

0

Tabs 4 through 13 consist of the actual source code listings for the benchmark program written in the target languages. Page numbers are found on each page in the lower right-hand corner because many of the listings continue beyond margin boundaries. Tab 1
Program DCT Benchmark
(Commented)

0

0

c()

~ (C

B-2

***	MICAUCOMPUTER MIGH ONDER LANGUAGES (MOL) PUR THE OCT APPLICATION.	- 1
***	CUMPUTER HOL FOR THE UCT. THE PHIMARY PURPOSE OF THE BENCHMARK	,
***	OBJECT PHUGRAM SIZE, AND TO ILLUSTRATE AUVANTAGES AND DISAUVANTAGES	,
***	THE BENCHMARK IS PRESENTED BELOW IN ALGURITHMIC FURM WHICH CLOSELY	
***	IU BE PHESENTED IN A PROGRAMMING DESIGN LANGUAGE. AT ANY HATE, THE	_
***	PHUGHAMMEN IN IMPLEMENTING THE BENCHMARK IN DIFFERENT LANGUAGES SUCH	
***	THE PROGRAMMER SHOULD READ THE FOLLOWING COMMENTED BENCHMARK CAREFULLY	
***	EXTENSIVE GUIDANCE IS GIVEN. ALSO, A SEPANATE DOCUMENT, ENTITLED	
	r	
***	TU CUDING THE BENCHMARK IN A PARTICULAR LANGUAGE.	_
	I'U CUDING THE BENCHMARK IN A PARTICULAR LANGUAGE. I'MIS BENCHMARK WAS DEVELOPED AT THE MARINE CORPS TACTICAL SYSTEMS SUPPORT ACTIVITY, CAMP PENDLETON, CALIFORNIA, 92855.	
***	IU CUDING THE BENCHMARK IN A PARTICULAR LANGUAGE. IMIS BENCHMARK WAS DEVELOPED AT THE MARINE CORPS TACTICAL SYSTEMS SUPPURT ACTIVITY, CAMP PENDLETON, CALIFORNIA, 92885.	•
***	IU CUDING THE BENCHMARK IN A PARTICULAR LANGUAGE. IMIS BENCHMARK WAS DEVELOPED AT THE MARINE CORPS TACTICAL SYSTEMS SUPPURT ACTIVITY, CAMP PENDLETON, CALIFORNIA, 92885.	
-	TU CUDING THE BENCHMARK IN A PARTICULAR LANGUAGE. IMIS BENCHMARK WAS DEVELOPED AT THE MARINE CORPS TACTICAL SYSTEMS SUPPURT ACTIVITY, CAMP PENDLETON, CALIFORNIA, 92055.	
***	CUNSTANT, TYPE, AND VANIABLE DECLARATIONS FOR DATA ITEMS HERDURED TO BE DELICAL SYSTEMS DELICALS OF CLARATIONS FOR DATA ITEMS HERDING TO BE SHOULD BE ACHIEVED BY USING PLITERALLY, REPLACE, THE PURCTIONAL OF THE SHOULD BE ACHIEVED BY USING PLITERALLY, REPLACE, THEANS, OR THE	*
-	TU CUDING THE BENCHMARK IN A PARTICULAR LANGUAGE. INIS BENCHMARK WAS DEVELOPED AT THE MAHINE CORPS TACTICAL SYSTEMS SUPPORT ACTIVITY, CAMP PENDLEION, CALIFORNIA, 92055.	*
	THE SENCHMARK WAS DEVELOPED AT THE MARINE COMPS TACTICAL SYSTEMS SUPPORT ACTIVITY, CAMP PENDLETON, CALIFORNIA, 92005. CUMSTANT, TYPE, AND VARIABLE DECLARATIONS FOR DATA ITEMS REQUIRED TO SE STUDBAL IN SCOPE, FOR LANGUAGES WHICH DO NOT SUPPORT TYPET, THIS FUNCTIONAL ACCIDENT. SHOULD BE ACHIEVED BY USING ILITERALLYT, IMEPLACET, IMEANST, OR THE LUGICAL EQUIVALENT.	*
	CUNSTANT, TYPE, AND VANIABLE DECLARATIONS FOR DATA ITEMS HERDURED TO BE DELICAL SYSTEMS DELICALS OF CLARATIONS FOR DATA ITEMS HERDING TO BE SHOULD BE ACHIEVED BY USING PLITERALLY, REPLACE, THE PURCTIONAL OF THE SHOULD BE ACHIEVED BY USING PLITERALLY, REPLACE, THEANS, OR THE	*
	TU CUDING THE SENCHMARK IN A PARTICULAR LANGUAGE. IMIS SENCHMARK HAS DEVELOPED AT THE MARLINE CORPS TACTICAL SYSTEMS SUPPORT ACTIVITY, CAMP PENDLETON, CALIFORNIA, 92058. CUMSIANT, TYPE, AND VANIABLE DECLARATIONS FOR DATA ITEMS REGULARD TO DE DEUBAL IN SCOPE. FOR LANGUAGES MHICH DO NOT SUPPORT TIPET, THIS FUNCTION SHOULD BE ACHIEVED BY USING PLITERALLY!, PREPLACE!, PREMIS!, OR THE LUGICAL EQUIVALENT.	*

: •	IIMINGCONTRUL = 277;	(* USED TO CONTRUL THE NUMBER OF TIMES THAT *) (* KERNEL , THE ACTUAL BENCHMARK EXECUTION '*) (* PROCEDURE IS CALLED. THIS AILL HE ADJUSTED*)
:	1UPONT = 7771	(+ SU THAT HUN TIMES ARE EASY TO MEASURE. +) (+ THIS PARAMETER IS INSTALLATION-DEPENDENT. +)
		(* IT WILL HE CHUSEN SO THAT OUTPUT OCCURS *) (* ON AN I/O PURT THAT IS NOT CONNECTED TO *) [* ANYTHING. *)
	NUMBERHSUS = 877	(CONTROLS THE NUMBER OF TIMES THE MESSAGE +)- (PRUCESSING LOUP IS EXECUTED. +)
o _	16318716 = 851	(+ AN EIGHT-GIT GUANTITY UF ALTERNATING GINARY+). (+ ONES AND ZEHOES TO TEST UUTPUT, SHIFTING,
: -	INTEGER 1 = 300;	(+ A 10-HIT INTEGER USED TO TEST INTEGER +) (+ A 10-HIT INTEGER USED TO TEST INTEGER +) (+ ARITHMETIC CAPABILITIES. +)
· -	INTEGERS # -15#1	(* A 10-BIT INTEGER USED TO TEST INTEGER *) (* ARITHMETIC CAPABILITIES. *)
· •	HLIMASK # 13	C+ USED TO MASK THE RIGHTMOST BIT IN A BYTE +1
3 -	M\$665MGTH = 801	(* INCUMING MESSAGES ARE EXPECTED TO BE 80 *) (* CHARACTERS IN LENGTH. *)
_	BUFFERMAA = 151	(+ ALL BUFFERS ARE 16 CHARACTERS IN LENGTH +) (+ RANGING FHOM & TO 15 INCLUSIVE +)
~ -	STARTCUDE . 'S'#	(4 THE ASCLE & DENUTES THE START OF A MESSAGE #)
٥)-		
	TYPE	
-	SUPPERIVE & APMAY (8, BU	(+ ALL CHARACTER BUFFERS HANGE FROM +) (+ W TO BUFFERMAX, [.E FROM & TO 15. +)
	VAR	
-	IIMINGLUDPEN, LUUPCOUNIEN, AMILECOUNIEN,	(* USED AS AN INUEX FOR THE TIMING LOUP +) (* USED AS A LOUP INDEX IN SEVERAL LOOPS. +) (* USED TO CONTRUL THE WHILE LOOP. +)
	UPHACTIUM: Indupfereth;	(* USED TO CUNIRUL THE CASE STATEMENT, *) (* HANGING IN VALUE FRUM & (0 9, *) (* INUEXES INTO THE INPUT BUFFER, INBUFFER, *)
, o _	UUTBUFFEHPTH : INTEGEH;	(* INDEXES INTO THE OUTPUT BUFFER, DUTBUFFER.*) (* ALL OF THESE & INTEGERS SHOULD BE IMPLE- +) (* MENTED AS ÎO-DIT INTEGERS. +)
_	анначі, аннач і банна	ARHAYSIZE) OF INTEGER; (+ BUTH ARMAYI AND ARMAYZ ARE 16-BIT INTEGER +)
ິ		(+ ARNAYS UF SIZE ARRAYSIZE (125) AND ARE +) (+ USED TO TEST 1-DIMENSIONAL ARRAY ACCESSING+)
	INSUPPER : SUPPERTYPE;	(* A CHARACTER BUFFER USED FUR INPUT +)
_	NEACHAR & CHAR!	(* A CHANACTER VARIABLE 113EU FOR TEMPORARY *)
·-	MIXELTYPE : RECORD;	(+ A COMPLEX MECURU (OR TABLE), CUMSISTING +)

.

CHARBUPPERIBUPPERITEE; C+ UP A 10 LIEM CHARACTER ARRAY FULLUMED BY INTHUMBER : INTEGERS (* A 16-BIT INTEGEN. FUR HECOMO WIXERLANDE? FUNCTION GETHYTE + CHAR! ************************************** 0 FUNCTION GETBYTE RETURNS A CHARACTER VALUE AMEN INVOKED. IT UBTAINS THE CHARACTER FRUM THE IN-PUT BUFFER (INDUFFER), AND INCREMENTS THE BUFFER INDEX (INBUFFEHPIH), CHECKING TO SEE IF IT EXCEEDS. BUFFERMAX IN SIZE. IF IT OUES, INSUFFERPTR IS RESET TO ZERO. NONE. INPUTSE JUTPUTS: GETBYTE, A CHAHACTER VALUE. CALLED BY! KERNEL. NU PHOCEDURES OR FUNCTIONS. CALLSE ******************** GETALE := INBUFFER [INBUFFERFIR] ; INSUFFERETH : INSUFFERETH + 1; IF INSUFFEREIR > SUFFERMAX THEN INSUFFEREIR := 4; ENU GETHYTE; PRUCEDURE PUTAYTE (INCHAR : CHAR, PUTBUFFER : SUFFERTYPE); PROCEDURE PUTBATE PUTS ONE CHARACTER INTO THE PHOLOGUES DUTPUT SUFFER (PUTSUFFER). IT THEN INCHEMENTS THE BUFFER INDEX (OUTBUFFERPTH), CHECKING TO SEE IF IT EXCEEDS BUFFERMAX IN SIZE. IF IT QUES, QUIBUFFEHPTR IS RESET TO LENG. INPUTSE INCHAR, A CHANACIER VALUE, AND PUTTUPFER, A CHARACTER BUFFER UF SIZE BUFFERMAX, MAICH IS PASSEU BY REFERENCE AS AN ARRAY TO TEST COMPILER EFFICIENCY IN PASSING MHULE ANNAYS. OUTPUTS: PUTBUFFER (CHANGED). CALLED BY: KEHNEL. CALLSE NO PRUCEDURES OR FUNCTIONS. BEGIN PUTBUFFER (QUTBUFFEHPTH) 12 INCHAH! O OUTBUFFERPIR IN OUTBUFFERPIR + 11 IF UUTHUFFEAPTR > BUFFERMAX THEN UUTHUFFEHPTR : # #1 ENU PUTETIES PHUCEBURE KERNELS

	 PROLOGUE: PROCEDURE KERNEL IS THE KEMNEL OF THE DCT 	•
	. BENCHMARK, IT IS CALLED HEPEATEDLY PHUM THE	*
	* MAIN PHOGRAM FROM A FOR LOUP CONTROLLED BY TIMING	-
	• LOOPEH, UPON INVOCATION, EXECUTION OF KERNEL	•
	PROCEEDS AS FOLLUMS: A NUMBER OF DUTPUTS ARE	<u>*</u>
)	* PERFURMED WITHIN A FUR LUUP, FULLUMED BY A NUMBER UF INTEGER ARRAY MANIPULATIONS, ALSO CONTROLLED D	
	A FOR LOUP. NEXT, A SENIAL DATA LINK IS SIMULATED	
	BY SEARCHING FUR A START COUE AND THEN INPUTTING	•
	. A 80-CHARACTER MESSAGE. THIS IS DUNE REPEATEDLY	•
	. AS CONTROLLED BY A FUR LUUP. THE NEXT LOOP SIM-	
	. ULATES OPERATUR INPUTS, ATTH MULTI-PATH BRANCHING	
	CONTRULLED BY THE CASE CUNSTRUCT, WITH THIS ACTION	
	* REPEATED A NUMBER OF TIMES SINCE THE LASE ACTION	
	THIS FOR LOOP, KERNEL RETURNS CONTRUL TO THE	-
	HAIN PHOGRAM.	•
 -	•	•
	* INPUTS: NONE.	•
	•	•
	+ UUTPUTS: NUNE.	•
	• PARISON MAN MARK MARK MARKET	*
	• CALLED BY: MAIN PROGRAM.	• —
	+ CALLS : FUNCTION GETWIE, PROCEDURE PUTBYIE.	•
	Andre to condition advantage conditions	•
	***************************************	*
	·	
	(+ LUCAL VARIABLE DECLARATIONS +)	
	VAM	
$\overline{}$	VAN OUTBUFFER : BUFFERTYPE; (> DEGLAMED LUCALLY	*} *}
)	VAM	*} *}
)	OUTBUFFER & BUFFERTYPE; (> DEGLARED LUCALLY (> TO PERMIT PASSING OF AN ARRAY TO	• }
) <u> </u>	OUTBUFFER & BUFFERTYPE; (> DEGLAMED LUCALLY (> TO PERMIT PASSING OF AN ARRAY TO (> PROCEDURE PUTBYTE.	*} *}
) -	OUTBUFFER : BUFFERTYPE; (> DEGLARED LUCALLY (* TO PERMIT PASSING OF AN ARRAY TO (* PROCEDURE PUTBYTE. 3EUIN (* PERFORM I/O OPERATIONS FOR IOLOUP NUMBER OF TIMES. NOTHING (* SHOULD BE CONNECTED TO LOPORT TO ELIMINATE PERIPHERAL DEVIC	•} •}
)	OUTBUFFER : BUFFERTYPE; (> DEGLARED LUCALLY (** TO PERMIT PASSING OF AN ARRAY TO (** PROCEDURE PUTBYTE. 3EUIN (** PERFORM I/O OPERATIONS FOR IOLOOP NUMBER OF TIMES. NOTHING (** SHOULD BE CONNECTED TO IOPORT TO ELIMINATE PERIPHERAL DEVICE (** TIMING DEPENDENCIES.	•} •}
)	OUTBUFFER & BUFFERTYPE; (> DEGLARED LUCALLY (** TO PERMIT PASSING OF AN ARRAY TO (** PROCEDURE PUTBYTE. 3EUIN (** PERFORM I/O OPERATIONS FOR IOLOOP NUMBER OF TIMES. NOTHING (** SHOULD BE CONNECTED TO LOPORT TO ELIMINATE PERIPHERAL DEVICE (** TIMING DEPENDENCIES. FUR LOOPCOUNTER 1** A TO IOLOOPS DU	•} •}
)	OUTBUFFER : BUFFERTYPE; (> DEGLARED LUCALLY (** TO PERMIT PASSING OF AN ARRAY TO (** PROCEDURE PUTBYTE. 3EUIN (** PERFORM I/O OPERATIONS FOR IOLOOP NUMBER OF TIMES. NOTHING (** SHOULD BE CONNECTED TO IOPORT TO ELIMINATE PERIPHERAL DEVICE (** TIMING DEPENDENCIES.	•} •}
	OUTBUFFER & BUFFERTYPE; (> DEGLARED LUCALLY (** TO PERMIT PASSING OF AN ARRAY TO (** PROCEDURE PUTBYTE. 3EUIN (** PERFORM I/O OPERATIONS FOR IOLOOP NUMBER OF TIMES. NOTHING (** SHOULD BE CONNECTED TO LOPORT TO ELIMINATE PERIPHERAL DEVICE (** TIMING DEPENDENCIES. FUR LOOPCOUNTER 1** A TO IOLOOPS DU	*) *) :
	OUTSUFFER (SUFFERTYPE: (DECLARED LUCALLY	*) *) :
	OUTBUFFER & BUFFERTYPE; (> DELLAMED LUCALLY (** TO PERMIT PASSING OF AN ARRAY TO (** PROCEDURE PUTBYTE. 3EUIN (** PERFORM I/O OPERATIONS FOR IOLOOP NUMBER OF TIMES. NOTHING (** SHOULD BE CONNECTED TO LOPORT TO ELIMINATE PERIPHERAL DEVIC (** TIMING DEPENDENCIES. FUR LOOPCOUNTER 1** A TO LOCOOPS DU DUTPUT(NOT(TESTBYTE), LOPORT); (** PERFORM INTEGER ARRAY OPERATIONS AS CONTROLLED BY ARRAYSIZE	*) *) :
)	OUTHUFFER & BUFFERTYPE; (* DECLARED LUCALLY (* TO PERMIT PASSING OF AN ARRAY TO (* PROCEDURE PUTBYTE. DEGIN (* PERFORM I/O OPERATIONS FOR LOUGH NUMBER OF TIMES. NOTHING (* SHOULD BE CONNECTED TO LOPORT TO ELIMINATE PERIPHERAL DEVIC (* TIMING DEPENDENCIES. FUR LOUPCOUNTER (* & TO LOUGHS DU OUTPUT (NOT (TESTBYTE), LOPORT); (* PERFORM INTEGER ARRAY OPERATIONS AS COMIRQUED BY ARRAYSIZE FOR LOUPCOUNTER (* & TO ARRAYSIZE DO ARRAY (LOUPCOUNTER); * ARRAYSIZE DO ARRAY (LOUPCOUNTER); * ARRAYSIZE DO	*) *) (b*) *) (b*) *)
	OUTSUFFER : SUFFERTYPE; (> DELLARED LUCALLY (* TO PEHMIT PASSING OF AN ARRAY TO (* PROCEDURE PUTSITE. SEGIN (* PERFORM I/O OPERATIONS FOR LOCOUP NUMBER OF TIMES. NOTHING (* SHOULD BE CONNECTED TO LOPORT TO ELIMINATE PERIPHERAL DEVIC (* TIMING DEPENDENCIES. FUR LOOPCOUNTER 1* 3 TO LOCOUPS DU DUTPUT(NOT(TESTBYTE), LOPORT); (* PERFORM INTEGER ARRAY OPERATIONS AS CONTROLLED BY ARRAYSIZE FOR LOOPCOUNTER 1* 0 TO ARRAYSIZE DO ARRAY1[COOPCOUNTER]; * ARRAY1]	*) *) (b*) *) *)
)	OUTSUFFER : SUFFERTYPE; (> DELLARED LUCALLY (> TO PERMIT PASSING OF AN ARRAY TO (> PROCEDURE PUTSYTE, SEGIN (* PERFORM I/O OPERATIONS FOR IDLOOP NUMBER OF TIMES, NOTHING (* SHOULD BE CONNECTED TO LOPORT TO ELIMINATE PERIPHERAL DEVIC (* TIMING DEPENDENCIES, FUR LOOPCOUNTER := 3 TO (OLOOPS DU OUTPUT(NOT(TESTSYTE), LOPORT); (* PERFORM INTEGER ARRAY OPERATIONS AS CONTROLLED BY ARRAYSIZE FOR LOOPCOUNTER := 0 TO ARRAYSIZE DO ARRAYSIZEONATER); (* SIMULATE PROCESSING AN INCOMING MESSAGE BY LOURING FOR A (* SIMULATE PROCESSING AN INCOMING MESSAGE BY LOURING FOR A	*) *) (b*) *) *) *) *) *)
	OUTBUFFER : BUFFERTYPE; (> DELLARED LUCALLY (* TO PERMIT PASSING OF AN ARRAY TO (* PROCEDURE PUTBYTE. (* PROCEDURE TO LOUGH NUMBER OF TIMES. NOTHING (* SHOULD BE CONNECTED TO LOUGHT TO ELIMINATE PERIPHERAL DEVIC (* TIMING DEPENDENCIES. FUR LOUPCOUNTER :* 3 TO LOUGHS DU OUTPUT(NOTITESTBYTE), LUPURT); (* PERFURM INTEGER ARRAY OPERATIONS AS CONTROLLED BY ARRAYSIZE FUR LOUPCOUNTER :* 3 TO ARRAYSIZE DO ARRAY [LOUPCOUNTER] :* ARRAY ZARRAYSIZE - LOUPCOUNTER); (* SIMULATE PROCESSING AN INCOMING MESSAGE BY LOURING FUR A (* SIART COPE; THEN INPUT MEGLENGIM NUMBER OF CHARACTERS (* USING LETHYTE TO INPUT EACH CHARACTER AND POTBYTE TO OUTPUT	*) *) (b*) *) *) *) *) *) *) *) *) *) *) *) *) *
)	OUTSUFFER : SUFFERTYPE; (> DELLARED LUCALLY (> TO PERMIT PASSING OF AN ARRAY TO (> PROCEDURE PUTSYTE, SEGIN (* PERFORM I/O OPERATIONS FOR IDLOOP NUMBER OF TIMES, NOTHING (* SHOULD BE CONNECTED TO LOPORT TO ELIMINATE PERIPHERAL DEVIC (* TIMING DEPENDENCIES, FUR LOOPCOUNTER := 3 TO (OLOOPS DU OUTPUT(NOT(TESTSYTE), LOPORT); (* PERFORM INTEGER ARRAY OPERATIONS AS CONTROLLED BY ARRAYSIZE FOR LOOPCOUNTER := 0 TO ARRAYSIZE DO ARRAYSIZEONATER); (* SIMULATE PROCESSING AN INCOMING MESSAGE BY LOURING FOR A (* SIMULATE PROCESSING AN INCOMING MESSAGE BY LOURING FOR A	*) *) (b*) *) *) *) *) *)
	OUTBUFFER : BUFFERTYPE; : DELLARED LUCALLY (* TO PERMIT PASSING OF AN ARRAY TO (* PROCEDURE PUTBYTE. (* PROCEDURE PUTBYTE. (* SHOULD BE CONNECTED TO LODGET TO ELIMINATE PERLIPHERAL DEVICE (* TIMING DEPENDENCIES. FUR LOOPCOUNTER :* A TO LOCOUPS DU OUTPUT(NOT(TESTBYTE), LUPURT); (* PERFORM INTEGER ARRAY OPERATIONS AS COMIRQUED BY ARRAYSIZE FUR LOOPCOUNTER :* O TO ARRAYSIZE DO ARRAY1[LOOPCOUNTER) :* ARRAY2[ARRAYSIZE - LOOPCOUNTER); (* SIMULATE PROCESSING AN INCOMING MESSAGE BY LOURING FOR A (* START CODE; THEN INPUT MEGLENGIN NUMBER OF CHARACTERS (* USING CETBYTE TO INPUT EACH CHARACTER AND POTBYTE TO OUTPUT (* EACH CHAPACTER, THIS MHOLE PROCESS IS MEREATED NUMBERNEGE (* NUMBER OF TIMES AS CONTROLLED BY THE FOR LOOP.	*) *) (b*) *) *) *) *) *) *) *) *) *) *) *) *) *
	OUTBUFFER : BUFFERTYPE; (* DECLARED LUCALLY (* TO PERMIT PASSING OF AN ARRAY TO (* PROCEDURE PUTBYTE. 356 IN (* PERFORM I/O OPERATIONS FOR LOUGH NUMBER OF TIMES. NOTHING (* SHOULD BE CONNECTED TO LOPORT TO ELIMINATE PERIPHERAL DEVIC (* TIMING DEPENDENCIES. FUR LOUPCOUNTER :* 3 TO LOUGHS DU OUTPUT(NOT(TESTBYTE), LOPORT); (* PERFORM INTEGER ARRAY OPERATIONS AS CONTROLLED BY ARRAYSIZE FUR LOUPCOUNTER :* 3 TO ARRAYSIZE DO ARRAY1[LTOPCOUNTER] :* ARRAY2[ARRAYSIZE - LOUPCOUNTER); (* STAULATE PROCESSING AN INCOMING MESSAGE BY LOURING FOR A (* START CODE; THEN INPUT MEGLENGIM NUMBER OF CHARACTERS (* USING LETHYTE TO INPUT EACH CHARACTER AND POTBYTE TO JUTPUT (* EACH CHAPACTER. THIS MOULE PROCESS IS MEPEATED NUMBERMSG (* NUMBER OF TIMES AS CONTROLLED BY THE FOR LOUP.	*) *) (b*) *) *) *) *) *) *) *) *) *) *) *) *) *
	OUTBUFFER : BUFFERTYPE; : DELLARED LUCALLY (* TO PERMIT PASSING OF AN ARRAY TO (* PROCEDURE PUTBYTE. (* PROCEDURE PUTBYTE. (* SHOULD BE CONNECTED TO LODGET TO ELIMINATE PERLIPHERAL DEVICE (* TIMING DEPENDENCIES. FUR LOOPCOUNTER :* A TO LOCOUPS DU OUTPUT(NOT(TESTBYTE), LUPURT); (* PERFORM INTEGER ARRAY OPERATIONS AS COMIRQUED BY ARRAYSIZE FUR LOOPCOUNTER :* O TO ARRAYSIZE DO ARRAY1[LOOPCOUNTER) :* ARRAY2[ARRAYSIZE - LOOPCOUNTER); (* SIMULATE PROCESSING AN INCOMING MESSAGE BY LOURING FOR A (* START CODE; THEN INPUT MEGLENGIN NUMBER OF CHARACTERS (* USING CETBYTE TO INPUT EACH CHARACTER AND POTBYTE TO OUTPUT (* EACH CHAPACTER, THIS MHOLE PROCESS IS MEREATED NUMBERNEGE (* NUMBER OF TIMES AS CONTROLLED BY THE FOR LOOP.	*) *) (b*) *) *) *) *) *) *) *) *) *) *) *) *) *
	QUIGUFFER : GUFFENTYPE; (* DELLANED LUCALLY (* TO PERMIT PASSING OF AN ARRAY TO (* PROCEDURE PUTBYTE. ** (* PROCEDURE PUTBYTE. (* PROCEDURE PUTBYTE. (* PROCEDURE PUTBYTE. (* PROCEDURE PUTBYTE. (* PROCEDURE IN TO LOUGH NUMBER OF TIMES. NOTHING (* SHOULD BE CONNECTED TO LOPORT TO ELIMINATE PERIPHERAL DEVIC (* TIMING DEPENDENCIES. FUR LOUPCOUNTER :* 3 TO LOUGHS DU OUTPUT(NOT(TESTBYTE), LUPURT); (* PERFORM INTEGER ARRAY OPERATIONS AS COMFROLLED BY ARRAYSIZE FUR LOUPCOUNTER :* 3 TO ARRAYSIZE DO ARRAY LUTTOROUNTER); * ARRAY SIZE DO ARRAY LUTTOROUNTER); * ARRAY SIZE DO (* SIANT COME; THEN INPUT MSGLENGIM NUMBER OF CHARACTERS (* USING LETBYTE TO INPUT LACH CHAMALTER AND POTBYTE TO JUTPUT (* SAME CHAPACTER, THIS ANDLE PROCESS IS MEPERTED NUMBERMSG (* NUMBER OF TIMES AS CONTROLLED BY THE FUR LOUP; FUR LOUPCOUNTER :* 8 TO NUMBERMSGS DU **REFORM :* GETBYTE; **NEFORM :* GETBYTE;	*) *) *) *) *) *) *) *) *) *) *) *) *) *
	OUTSUFFER : SUFFERTYPE; (* DELLARED LUCALLY (* TO PERMIT PASSING OF AN ARRAY TO (* PROCEDURE PUTSTTE. DEGIN (* PROCEDURE PUTSTTE. (* PROCEDURE PUTSTTE. (* PROCEDURE PUTSTTE. (* PROCEDURE PUTSTTE. (* PROCEDURE I/O OPERATIONS FOR LOLOUP NUMBER OF TIMES. NOTHING (* SHOULD BE CONNECTED TO LOPORT TO ELIMINATE PERTPHERAL DEVIC (* TIMING DEPENDENCIES. FUR LOUPCOUNTER 1= 3 TO LOLOUPS DU OUTPUT(NOT(TESTSYTE), LUPURT); (* PERFORM INTEGER ARRAY OPERATIONS AS CONFROLLED BY ARRAYSIZE ARRAY1[LOUPCOUNTER] 1= 3 TO ARRAYSIZE DO ARRAY1[LOUPCOUNTER] 1= 4 ARRAYSIZE DO (* SIMULATE PROCESSING AN INCUMING MESSAGE BY LUBRING FOR A (* SIART CODE; THEN INPUT MEGLENGIM NUMBER OF CHARACTERS (* USING LETHYTE TO INPUT MEGLENGIM NUMBER AND POTSTET TO DUTPUT (* RACH CHAPACTER. THIS MOULE PROCESS IS MERETAD NUMBERMSG (* NUMBER OF TIMES AS CONTROLLED BY THE FUR LOUP. FUR LOUPCOUNTER 1= 8 TO NUMBERMSGS DU MEPCRAR 1= 8 TO NUMBERMSGS DU NECCHAR 1= 6ETSYTE; UNTIL MERCHAR = STARTCODE; BEGIN NECCHAR 1= 6ETSYTE;	*) *) *) *) *) *) *) *) *) *) *) *) *) *
	QUIGUFFER : GUFFENTYPE; (* DELLANED LUCALLY (* TO PERMIT PASSING OF AN ARRAY TO (* PROCEDURE PUTBYTE. ** (* PROCEDURE PUTBYTE. (* PROCEDURE PUTBYTE. (* PROCEDURE PUTBYTE. (* PROCEDURE PUTBYTE. (* PROCEDURE IN TO LOUGH NUMBER OF TIMES. NOTHING (* SHOULD BE CONNECTED TO LOPORT TO ELIMINATE PERIPHERAL DEVIC (* TIMING DEPENDENCIES. FUR LOUPCOUNTER :* 3 TO LOUGHS DU OUTPUT(NOT(TESTBYTE), LUPURT); (* PERFORM INTEGER ARRAY OPERATIONS AS COMFROLLED BY ARRAYSIZE FUR LOUPCOUNTER :* 3 TO ARRAYSIZE DO ARRAY LUTTOROUNTER); * ARRAY SIZE DO ARRAY LUTTOROUNTER); * ARRAY SIZE DO (* SIANT COME; THEN INPUT MSGLENGIM NUMBER OF CHARACTERS (* USING LETBYTE TO INPUT LACH CHAMALTER AND POTBYTE TO JUTPUT (* SAME CHAPACTER, THIS ANDLE PROCESS IS MEPERTED NUMBERMSG (* NUMBER OF TIMES AS CONTROLLED BY THE FUR LOUP; FUR LOUPCOUNTER :* 8 TO NUMBERMSGS DU **REFORM :* GETBYTE; **NEFORM :* GETBYTE;	*) *) *) *) *) *) *) *) *) *) *) *) *) *

```
WHILE AMILECUUNTER < MOULENGTH DO
0
                          ( * THE PURPOSE OF THIS MHILE LUOP IS TO TEST A LANGUAGE S+)
                          ( * SUPPORT 4 WHILE CUNSTRUCT. THE WHILE LUGIC MUST BE
                          ( * IMPLEMENTED BY THE STRUCTURED USE OF A GUTO OR OTHER
                          ( COMSTRUCT. ITERATIVE LUUPING IS NOT PERMITTED.
                             BEGIN (+ INPUT MESSAGE CHANACIENS +)
Ü
                                NEMCHAN : & GETBYTES (+ GET NEXT CHANALTEN IN MSG
                                 (* NOW SHIFT THE CHARACTER HIGHT UNE BIT AND SEE IF+)
                                 ( * THE HIGHTHUST BIT IS A ONE.
                                IF (MIGHISHIFT (NEWCHAR, 1) AND BITMASK) = 1 THEN
                                    PUTBYTE (NEACHAR, UUTBUFFER);
                                ELSE ( + DO THE SAME THING - THIS IF/ELSE IS TO
                                                                                        +}
                                      C+ CHECK CUMPILER IF/ELSE CAPABILITY ONLY.
                                    PUTHYTE (NEWCHAR, CUISUFFER) !
                                AMILECOUNTER 18 AMILECOUNTER + 13
                             END MHILES
()
                       ENU: (+ FOR LUUPCOUNTER 1= # TU NUMBERMSGS LOUP +)
                                    ( INITIALIZE UPRACTIUN TO ZERO SU IMAT IT CAN
                UPHACTION := 0;
                                    (+ HANGE IN VALUE PHUM W TU 9. IN INUSE
                                    ( LANGUAGES WHICH SUPPORT A MODULU FUNCTION,
                                                                                        4 }
                                    ( * OPRACTION SHOULD SE IMPLEMENTED AS À MODULO
                                    C+ UTHERWISE IT SHUULD BE INCREMENTED BY UNE
                                    C+ EACH LUUP, CHECKEU AGAINST 9, AND RESET TO
                                                                                        + }
                                    ( - ZERO IF GREATER THAN Y AS DONE BELOW AT THE
                                                                                        + )
                                    ( - BUTTOM UP THE CASE STATEMENT.
                FUH LOOPCOUNTER 18 & TO UPRACTIONLOOPS DO BEGIN (+ CASE UPHACTION WHERE UPHACTION HANGES FROM & TO 9
                       CASE UPRACTION OF
                                       L+ SHIFT TESTBYTE IN THE RIGHT OF THREE SITS
                                       C. THEN STURE IN MIXEUTYPE.
                                       MIXEDTYPE CHARBUPFER (UPHACTION) :=
                                          HIGHTSHIFT (TESTETTE, 3);
\bigcirc
                                       (+ CIRCULAN MUTATE TESTATTE TO THE LEFT BY 2
                                       L+ BIT IS A UNE. STURE IN MIXEUTYPE.
                                                                                        • )
                                       MIXEDTYPE CHARBUFFER LUPHACTIUNI 1.
                                          LEFTRUTATE (TESTEYTE, 2) AND BLIMASK;
 Э
                                       (+ MOVE ALL 16 CHARACTERS IN INSUFFER TO +)
(+ THE CHARACTER SUFFER PORTION UP MIXEDTYPE +)
                                       MIXEDTYPE. CHARBUFFER IS INSUFFER!
                                       C+ WILL BE EXECUTED WHEN OPRACTION . 5 UH d.
                          OTHERWISE:
                                       L+ TEST LANGUAGE CAPABILITY FOR 10-ALT
 \circ
                                       ( * AMITHMETIC. * TARENTY * (((INTEGERIZINTEGERS) *
                                          INTEGER11/INTEGER2) + INTEGER1;
                       ENU CASE!
 0
                                                       ( + INCHEMENT OPHALTION AND ( + ALLUM IT TO HANGE FROM &
                       OPPACTION IN OPPACTION + 13
                       IF UPRACTION > 9 THEN
                                                        ( IU 9 IN VALUE.
                                                                                        • )
                          UPHACTION := 11
                    ENDS (+ POH LOUPCOUNTER 1= & TO UPHALITUNLUOPS +)
 O
               ENU! ( PRUCEDINE KENNEL +)
```

(+ MAIN PROGNAM EXECUTION BE (+)	Rena unit	
(********************		*} *)_
	*************	*****
MHITELINE (BEGIN BENCHMARK EXECUTION)); (+ 4HEN THIS MESSAGE A	PPEARS
	I+ UN THE CRIP BEGIN T	IMING
	I THE BENCHMARK EXECU	TIUN.
AHRAY2 (LOUPCOUNTER) := LOUPCOUNTER;	(FILL ARRAY2 MITH ASCE	NO ING
	(+ INTEGERS, W - 15.	
FUR INSUFFERENT := & TO SUFFERMAN DU		
INDUFFER [INBUFFEMPTH] 12 'A']	(+ FILL THE INPUT SUFFER	WITH
• •	(PHUNY TEXT , ALL AIS.	
THE RECUALITY OF STATEFORM	. H. T ONE STANTIONS IN	
INDUPPER (10) := STARTCOUE;	(* Pui une staktione in (* 30 kehnel *ill find	
	The second secon	
INCUFFEREIR, OUTCOFFEREIR 1= 31	ONI REMOUDE HTDE TRIDE +)	
	(TU FIRST CHARACTER IN	HUFFE
FOR TIMINGLOOPER := 8 TO TIMINGCONTROL	DO (+ THIS LOOP CONTROLS	HUA
KÉRNELI	(* MANY TIMES KERNEL	IS
	(- CALLED, THUS CONTR	
	(+ BENCHMARA EXECUTIO	u ITUE
AMIILLINE (MIXEDIYPE, 'END EXECUTION')	(+ AMEN THIS MESSAGE AP	PEARS (
	(THE CRT, STUP TIMING	THE
	(BENCHMARK EXECUTION	TIME.
NU.		
		
		·
· ·		
		·
		B-8
		B-8

ية الأسياد التستي الأدار التيار الشركة التواليات

Tab 2
Program DCT Benchmark (Code Only)

C·

0

0()

 \circ

0

0

3-9

```
PROGRAM UCTBENCHMARK
              CUNSI
                                               IULUUPS = 5/5;
                 ANHAYSIZE = 125;
                                               TIMINGCONTHUL = 371
                 UPHACTIONLOUPS - 100;
                 LOPORT = 277
                                               NUMBERMSGS = 2001
                 TESTUYTE = 851
                                               INTEGERÎ = JUUI
                  INTEGERS = -154;
                                               MITMASK # 1;
                                               BUFFERMAX = 15;
                  MSGLENGTH = 801
                 STARICQUE = 15';
              TYPE
                 SUPPERTYPE . ARRAY (0. JUFFERMAX) OF CHAR!
              VAR
0
                  FIMINGLUOPER, LOUPCOUNTER, WHILECOUNTER, OPRACTION, INSUFFERPTR,
                    QUIBUFFERPT# : INTEGER:
                  ARRAYI, ARRAYZ : ARRAYIJ., ARRAYSIZEJ UP INTEGERI
                  INSUFFER : SUFFERTYPE;
                 NEACHAR : CHAR!
                 MIXEDTYPE : RECORDS
                    CHARBUFFER : BUFFERTYPE;
INTNUMBER : INTEGER;
                 END RECURD MIXEDTYPE;
               FUNCTION GETBYTE : CHAMI
( ;
                  BEGIN
                     GETHYTE := INGUFFER (INGUFFERPTR);
                     INDUPPERPTH := INBUPFERPTH + 11
                     IF INBUFFEROTH > SUFFERMAX THEN INSUFFEROTH := 0;
                  END GETAYTE!
              PRUCEDURE PUTHYTE (INCHAR : CHAR, PUTBUFFER : BUFFERTYPE);
                  BEGIN
                     PUTBUFFER (UUTBUFFERPTR) := INCHAR)
                     JUTAUPPENPTP := QUTAUPPENPTR + 1;
                     IF OUTBUFFEMPIR > BUFFERMAX THEM OUTBUFFERPTR := ##
                  END PUTHTES
0
              PHUCEBURE KERNELS
                  VAH OUTSUPFER : SUFFERTYPE!
                  BEGIN
                     FOR LOOPCOUNTER IS 3 TO TOLOOPS DO
                        QUIPUT(NOT(TESTEYTE), LUPORT);
                     FUR LUOPCOUNTER 18 9 TO ARRAYSIZE DU
                        ANNAYI (LOUPCOUNTER! 1= ARNAYZ (ANNAYSIZE - LOUPCOUNTER);
Ü
                     FOR LOUPCOUNTER 18 & TO NUMBERMSGS DU
                        DEGIN
                           REPEAT
                                                                              B-10
                              NEWLMAN IS GETSYTE!
                           UNTIL NEACHAR & STARICULES
```

WHITECOUNTER IS WI

##ILECQUATER < MSGLENGIN DU ##EGIN NEWCHAR IS GETSTIE; IF (MIGHTSHIFT(NEWCHAR, 1) AND BITMADA) = 1 THEN PUTSTIENESCHAR, DUTSDFFER)) ELSE PUTSTIENESCHAR, DUTSDFFER); END ***ILES END ***ILES END ***ILES PUTSTIENESCHARA, GUTSDFFER); END ***ILES END ***ILES ***ILECQUATER IS ***ILECQUATER ***I) END ***ILES END ***ILES ***ILECQUATER ***INTEQUATER ***I) ***INTEQUATER	auti s	will acousted a Madrida Communication
IF (AIGHTSHIFT(NEMCHAM, 1) AND BITMASH; = 1 THEN PUTSTTE(NEMCHAM, DUTSUFERR); END PUTSTE(NEMCHAM, DUTSUFERR); PUTSTE(NEMCHAM, DUTSUFERR); PUTSTE(NEMCHAM, DUTSUFERR); PUTSUFE(NEMCHAM, DUTSUFERR); PUTSUFE(NEMCHAM, DUTSUFERR); PUM LOOPCOUNTED := 4 TO OPRACTIUNLOOPS DO SEGIM CASE OPHACTION OF CASE OPHACTION OF 1,4,7 : MIREDTYPE, CHAMBUFFEM (OPMACTION) := LEFTHOTATE(TESTATTE, 2) AND OILMASH; 2,0 : MIREDTYPE, CHAMBUFFEM := NOBUFFEM; LEFTHOTATE(TESTATTE, 2) AND OILMASH; 2,0 : MIREDTYPE, CHAMBUFFEM := ((((INTEGEM!/INTEGEM2) + INTEGEM!)) END CASE; OTHEMMISE: MIREDTYPE, INTINUMBEM := ((((INTEGEM!/INTEGEM2) + INTEGEM!)) END CASE; OPHACTION := OPHACTION + := 0; END FOR: SEGIN (= PROGRAM EXECUTION +) ANTICLIME ("SEGIN SENCHMAMA EXECUTION);; FUM LOUPCOUNTEM := 0 TO SUMPERMAX DU AMATZILOOPCOUNTEM; := 0 TO SUMPERMAX DU INSUPPEM(INSUPFEMPTM) := 10; FUM INSUPPEM(INSUPFEMPTM) := 10; FUM ISSUPPEM(INSUPFEMPTM) := 0; FUM ISSUPPEM(INSUPFEMPTM) := 0		GIN
DUTSTIE (NEACHAR, OUTSUFFER)) ***MILECOUNTER 18 ***MILECOUNTER ***) END ***LEE) END ***LEE) ***DIM LOOPCOUNTER 18 ***MILECOUNTER *** ***PUM LOOPCOUNTER 18 ***ID OPRACTIUNLOOPS UD ***BEGIN CASE OPHACTION OF 1,4,7		IF (RIGHTSHIFT(NEWCHAR, 1) AND BITMASK) # 1 THEN
END FORE END FORE UPHACTION IS 33 PUM LODPCOUNTED IS 4 TO OPPACTIONLOOPS DO SEGIN CASE OPHACTION OF I,4,7 MIXEDTYPE, CHARBUFFER (OPMACTION) IS AIGHTSHIFT(TESTBYTE, 3); I,4,7 MIXEDTYPE, CHARBUFFER (OPMACTION) IS LEFTHOTATE(TESTBYTE, 2) AND OLIMASK; C,0 MIXEDTYPE, CHARBUFFER IS INSUFFER; OTHERWISE; MIXEDTYPE, INTINUMBER IS ((((INTEGEN!/INTEGEN2))) END CASE; OPMACTION IS OPHACTION + 1; IF UPMACTION > 9 THEN OPPACTION IS 0; END FORM SEGIN (= PROGRAM EXECUTION *) AHLIBLINE (=SEGIN SENCHMARK EXECUTION); FUN LUMPCOUNTER IS 0 TO ARRAYSIZE OO ARRAYZILOOPCOUNTER] IS LOOPCOUNTER; FUN LUMPFERMIN IS 3 TO SUFFERMAX DU INSUFFERMIN OUTSUFFERMIN IS 0; FUN INSUFFERMIN OUTSUFFERMIN IS 0; FUN ITMINUSUPPER IS 0 TO TIMINGCONTROL DU ARRIELINE (MIXEDTYPE, 'END EXECUTION'); END, END, END, END, ARLIELINE (MIXEDTYPE, 'END EXECUTION');		ELSE
DPMACTION 1= 01 FUN LOOPCOUNTEP 1= 0 TO OPRACTIUNLOOPS DO SECIN CASE OPMACTION OF 1,4,7	EM()	
FUN LOOPCOUNTER 18 4 TO OPRACTIUNLOOPS UD SEGIN CASE OPRACTION OF 3,3,9 : **ILEDTYPE, CHARBUFFER (OPRACTION) := **IGHTSHIF (TESTBYTE, 3); 1,4,7 : **ILEDTYPE, CHARBUFFER (OPRACTION) := **IGHTSHIF (TESTBYTE, 2) AND OLIMASK; 1,4,7 : **ILEDTYPE, CHARBUFFER (OPRACTION) := **LEFTHOTATE(TESTBYTE, 2) AND OLIMASK; 2,0 : **ILEDTYPE, CHARBUFFER := (NBUFFER) **OTHERWISE: **ILEDTYPE, INTINUMBER != (CCC(INTEGENT/INTEGENZ) **INTEGENT) /*INTEGENZ) **INTEGENZ) ** INTEGENT; **END CASE; **OPRACTION := OPRACTION * 1; **END FUN; **END **UN; **INTEGENZ := OPRACTION := OPRACTION := OPRACTION := OPPACTION := OPPACTIO		- - - -
SEGIN CASE OPHACTION OF ###################################	UPHACTION 18 0	5;
CASE OPHACTION OF ###################################		P 1 # 4 TO OPRACTIUNLOUPS UG
I,4.7 : MIXEDTYPE, CHAMBUPFER (UPRACTION) := LEFTHOTATE (TESTATTE, 2) AND SLIMASK; 2,0 : MIXEDTYPE, CHAMBUFFER; := (NOUFFER); OTMERMISE: MIXEDTYPE, INTNUMBER := ((((INTEGER!/INTEGER2) * INTEGER!)/INTEGER2) * INTEGER!) END CASE; OPHACTION := OPHACTION * 1; END HOR; END HOR; END HOR; END KERNEL; SEGIN (* PRUGNAM EXECUTION *) #HIGHIME (:BEGIN SENCHMANN EXECUTION:); FUN LUMPCOUNTER: := 0 TU ARRAYSIZE UU ARRAYZILOOPCOUNTER: := 0 TU SUPFERMAX UU INSUFFERMTH := 2 TU SUPFERMAX UU INSUFFERMTH := 2 TU SUPFERMAX UU INSUFFERMTH, OUTSUFFERPTH := 0; FUN INSUFFERMTH, OUTSUFFERPTH := 0; FUN IIMINGLOUPER := 0 TU TIMINGCUNTHUL UU AEMNEL; ANLIELINE (MIXEDTYPE, 'END EXECUTION'); END.		PACTION OF
LEFTROTATE(TESTBÝTE, 2) AND GI_MASK; 2,0 : MIXEDTYPE_CHARGUFPEM;: INBUFFEM; OTHERWISE: MIXEDTYPE_INTNUMBER := ((((INTEGEM!/INTEGEM2) * INTEGEM!) END CASE; OPHACTIOM := OPHACTION + 1;	0,5,5	
OTHERWISE: MIXEDTYPE, INTNUMBER := ((((INTEGER!/\INTEGER) INTEGER!/\INTEGER?) * INTEGER!/ END CASE; OPHACTION := OPHACTION + 1; END POUR; END POUR; END FOUR; END WERNEL; SEGIN (* PROGRAM EXECUTION *) ANTICLINE ('MEGIN MENCHMANN EXECUTION); FUN LOUNCOUNTER := 0 TO ARRAYSIZE DO ANNAYZILOOPCOUNTER! := LOUPCOUNTER; FUN INBUFFERNIR := 3 TO MUFFERMAN DO INBUFFER(INBUFFERPIN := 'A'; INBUFFER(IN) := STANTCODE; INBUFFER(IN) := STANTCODE; FUN IIMINGLOOPEN := 3 TO TIMINGCONTHOL DO ACHNEL; ANTICLINE (MIXEDTYPE, 'END EXECUTION'); END,	1,4,7	
INTEGER() / INTEGER() * INTEGER() END CASE) OPHACTION := OPHACTION + 1; IF OPHACTION > 9 THEN OPRACTION := 0; END FOR; END FOR; END FOR; SHOU KERNEL; JEGIN (* PROGRAM EXECUTION *) ANTICLINE (**JEGIN SENCHMANN EXECUTION); FOR LOUPCOUNTEN := 0 TO ARRAYSIZE DO ARRAYZICOOPCOUNTEN; FOR INBUFFEMPTH := 0 TO SUPFERMAN DO INBUFFEMPTH) := 10 SUPFERMAN DO INBUFFEMPTH) := 10; INBUFFEM [10] := STANTCOOE; INBUFFEM [10] := STANTCOOE; FOR ILMINGLOOPER := 0 TO TIMINGCONTROL DO ARRAYEL; ANTICLINE (MIXEDTYPE, 'END EXECUTION'); END.	2,0	: MIXEDTYPE.CHARBUFFER ;= INBUFFER;
OPHACTION := OPHACTION + 1; IF UPRACTION > 9 THEN OPRACTION := 0; END FUR; END REMEL; EGIN (* PROGRAM EXECUTION *) ***HITELINE ('SEGIN SENCHMARK EXECUTION); FUR LOUPCOUNTER := 0 TO ARRAYSIZE OO ARRAYZILOOPCOUNTER; := LOOPCOUNTER; FUR IMBUFFERPTR := 3 TO SUPFERMAX DO INSUPPER(IS) := STARTCODE; INSUPPER(IS) := STARTCODE; INSUPPERPIN, OUTSUPFERPTR := 0; FUR ILMINGLOUPER := 0 TO TIMINGCONTROL DO ARRAEL; ***HITELINE (MIXEDTYPE, 'END EXECUTION'); NO.	OTHER	(SREDETALVINEDERINE)))) =: HERMININI. ERFLERINE : BELWININI
IF UPRACTION > 9 THEN UPRACTION 1 ** WI END FUR; END XERREL; SEGIN (* PRUGHAM EXECUTION *) **********************************	END CASE	·
END FUR; END KERNEL; SEGIN (* PROGRAM EXECUTION *) WHITELINE (*BEGIN BENCHMARK EXECUTION); FUR LOUPCOUNTER; ** 0 TO ARRAYSIZE OD ARRAYZILOOPCOUNTER; ** LOOPCOUNTER; FUR INBUFFERPTR; ** 0 TO BUFFERMAX DO INBUFFER(INBUFFERPTR) ** 1**; INBUFFER(IB) ** STARTCODE; INBUFFER(IB) ** STARTCODE; FUR IMINGLOOPER ** 0 TO TIMINGCONTROL DO AERNEL; WHITELINE (MIXEDTYPE, *END EXECUTION*); END.		
#HITELINE ('BEGIN BENCHMARK EXECUTION'); FUR LUUPCOUNTER; = 0 TO ARRAYSIZE DO ARRAYZILOOPCOUNTER; = LOUPCOUNTER; FUR INBUFFERPTR := 0 TO BUFFERMAX DU INBUFFER(INBUFFERPTR) := 'A'; INBUFFER(Id) := STARTCODE; INBUFFER(Id) := 0 TO TIMINGCONTROL DU AERNEL; #HITELINE (MIXEDTYPE, 'END EXECUTION'); END.	END FUR;	
FUN LUUPCOUNTER := 0 TU ARRAYSIZE UU ARRAYZILOOPCOUNTER: := LOUPCUUNIEN; FUN LUBUFFERPIR := 3 TU SUPFERMAX DU INBUFFER [18] := STARTCOUE; INBUFFERPIR, OUTBUFFERPIR := 0; FUN IIMINGLOOPER := 3 TU TIMINGCUNTRUL DU AERNEL: ANTIELINE (MIXEDTYPE, 'END EXECUTION'); END.	SEGIN (+ PRUGHAM EA	(ECUTION +)
ARMAT2 LOGPCOUNTER: 18 LOGPCOUNTER: FUM INBUFFEMPTR : 2 TO SUPPERMAN DU INBUPPER [18] : 2 STARTCODE: INBUPPERPTM, OUTBUPPERPTH : 2 8; FUM IIMINGLOUPER : 2 0 TO TIMINGCONTROL DU ABANEL: MMIIELINE (MIXEDTYPE, 'END EXECUTION'); END.	MATIFFINE (196819	SENCHMARK EXECUTION:);
INSUPPER(INSUPPERPTR) := 'A'; INSUPPER(IS) := STARTCODE; INSUPPERPTR, OUTSUPPERPTR := 8; FUR IIMINGLOUPER := S IU TIMINGCUNTROL DU ABANEL; ANTIBLINE (MIXEDTYPE, 'END EXECUTION'); END,		
INBUPPER(18) := STARTCODE; INBUPPERPTH, OUTBUPPERPTH := 0; FUH IIMINGLOUPER := 0 TO TIMINGCONTHOL OU ARMEL; ANTIBLINE (MIXEDTYPE, 'END EXECUTION'); END.		
FUN TIMINGLOUPEN := 0 TO TIMINGCUNTHUL DU AEMNEL: ANTIELINE (MIXEDTYPE, 'END EXECUTION'); END.		
MATIFFINE (WIXEDIABE, JEND EXECUTION.))	FUN TIMINGLOUPER	,
		TYPE, 'END EXECUTION');
B-11	ÈNO,	
B-11		
B-11		
B-11		
		5-11
		د در چه همین در است. است ای این در

Tab 3 Error-Seeded DCT BENCHMARK

O

Ö

0

٥

Q

0

3

```
C
         PROGRAM UCTBENCHMARK
                                                              BENCHMARK
         ...
                     ERRUR
                                    SEEDED
                                                    UCT
                                                                                           ***
                                                                                           ...
         ...
                THIS VEHSION OF THE DCT BENCHMARK CONTAINS FIVE ERRURS DESIGNED
         ...
                                                                                           • • •
               TO CAUSE COMPILE-TIME ERRORS DUE TO INCURRECT SYNTAX. MOMEVER, ***
NOT ALL COMPILERS WILL RECOGNIZE ALL OF THE ERRORS AT COMPILE TIME.***
THE FIVE ERRORS ARE CLEARLY MARKED IN THE LISTING BELOW WITH ***
         ...
***
               A SERIES OF ASTERISKS ALONG WITH AN EXPLANATION OF THE EHRON.
         ***
                                                                                           * • •
                                                                                           * * *
         ***********************************
0
            CUNST
2
                ARRAYSIZE = 12#1
                                               IULUUPS # 575;
               UPHACTIUNLOUPS = 1881
                                               TIMINGCUNTHÚL # 77;
               TOBORT # 331
                                               NUMBERMSGS = 2001
                16575776 = 05;
                                               INTEGERÍ = 300;
                                               SITMASK = 1;
                INTEGERE = -15"
               MUSELENGTH = 861
                                               BUFFERMAX = 151
                STARTCODE = 'S';
            TYPE
                DUFFERTYPE & APRAY (W. . HUFFERMAX) UF CHAR;
 0
            VAR
            (****
                      ERROR 1: BY PLACING CUMMENT BRACKETS AROUND LOUPCOUNTER IN THE +)
                         LINE BELOW, THE DECLARATION OF LOUPCOUNTER IS ELIMINATED.
                                                                                             + )
            ( ****
                          THE PURPOSE IS TO SEE MOW DIFFERENT COMPILERS TREAT
                          VARIABLE DECLARATIONS OR THE LACK THEREUF.
            (+--+
.0
                IIMINGLUUPER, ( * LOUPCUUNTER, *) WMILECUUNTER, OPRACTIUN, INDUPFERPTR,
                   UUTSUPFERPIS : INTEGER;
                AMMAY, AMMAY2 : AMMAY (0 . AMMAYSIZE) UF INTEGERS
                INBUFFER : BUFFERTYPE;
                NEMCHAR : CHAR!
                MITEDIALF : MFCCHO!
                   CHARBUPFER : BUFFERTYPE;
                   INTNUMBER : INTEGERS
                CHU RECORD MIXEDIANET
            FUNCTION GETHYTE : CHAR!
                DEGIN
                   GETSYTE := !NOUFFER (INSUFFERPTH);
                   INSUPPEMENT IS INSUPPEMENT + 1;
IF INSUPPEMENT > SUFFERMAX THEN INSUPPEMENT IS 3;
                ENU GETHYTE;
 O
```

	PROGEOURE PUIDTIE LINGRAM I GMAM, PUIDUPPEM I DUFFERTYPEJI Degin
_	PUTBUFFER (OUTBUFFERPTR) := INCHAR; OUTBUFFERPTP := OUTBUFFERPTR + 1;
, () _	(++++ ERROR 2: IN THE LINE BELOW, BUFFERMAN HAS BEEN REPLACED +) (+++++ WITH NEWCHAR SU THAT JUTBUFFERPTR, AN INTEGER, IS +) (+++++ NUW CUMPARED TU NEWCHAR, A CHARACTER WUANTITY. THE +) (+++++ PURPOSE IS TO EVALUATE TYPE-CHECKING OF CUMPARISUMS. +)
	IF OUTBUFFERRY > NEWCHAR THEN OUTBUFFERRIN := 8; END PUTBYTE;
	PHUCEDURE KERNEL;
	VAN UUTSUFFER 1 SUFFERTYPER
_	BEGIN
, -	FOR LOUPCOUNTER := 3 TO IOLDOPS DO OUTPUT(NOT(TESTBYTE),IOPONT);
_	FOR LOOPCOUNTER : # TO ARHAYSIZE DU ARHAY: TENTOUDOPCOUNTER: : ARHAYSIZE - LUUPCOUNTER: ;
	FUR LOUPCOUNTER := 0 TO NUMBERMSGS DU BEGIN
-	REPEAT NEWCHAR := GETBYTE;
	UNTIL NEWCHAR & STARTCOUE;
	WHILECGUNTER := 0;
\cap	MHILE MHILECUUNTER « MAGLENGTH DO
· /	HEGIN NEMCHAR := GETHYTE:
_	(***** ERRUR 3: IN THE LINE BELUM, UUTBUFFEH HAS *) (***** SEEN HEPLACED WITH ARRAY: THE PUHPUSE *)
	(***** IS TO EVALUATE TYPE-CHECKING OF PARA- *) (***** METERS BEING PASSED, SINCE ARRAY IS *) (***** AN INTEGER ARRAY AND OUTSUPFER IS A *)
	(++++ CMARAGTER ARRAY. +)
-	IF (RIGHTSHIFT(NEMCHAR, 1) AND BITMASK) # 1 THEN PUTBYTE(NEMCHAR, AHRAYI);
	ELSE PUTBYTE(NEWCHAR, OUTBUFFER);
_	WHILECOUNTER : WHILECOUNTER + 1;
_	(**** ERRUR 4: THIS ERRUR INVULVES REMUVING AN "ENU" *) (**** BY GHACKETING IT AS A CUMMENT. THE PURPUSE IS *)
,	(+++++ TO EVALUATE EHRUR HECUVERY WITH MISSING +)
_	(+++++ BEGIN/END PAIRS, IN LANGUAGES WITHOUT BEGIN/ +)
	(* * * * * SUPFICE AS THE LOGICAL EQUIVALENT. *) (* END WMILE; *)
_	END FUR!
)	UPHACTIUM := 61
\circ	FOR LOUPCOUNTER IS & TO OPPACITUALOUPS NO B-14 BEGIN
~	CASE OPHACTION OF
•	0.3.9 : MIXEDTYPE.CMANSUPPER[UPRACTION] :=
O	U.J.9 : MIXEDTYPE.CMANBUPPER (UPRACTION) :#

....

	والمراب والمراب والمراب والمراب والمعالية والمراب والمعالية والمراب والمراب والمراب والمراب والمراب والمراب والمراب
	RIGHTSMIFT(TESTOTTE, 3);
C	1,4,7 : MIXEUTYPE.CHARBUFFER(OPRACTION) :=
	LEFTHOTATELTESTETTE, 2) AND BLIMASK;
(3)	2,0 ! MIXEDTYPE, CHARBUFFEH : * INBUFFEH;
\neg	(+++++ ERHOR 5: IN THE NEXT LINE, A LEFT MARENTHESIS +)
~	(+++++ HAS BEEN REMOVED TO CHECK FOR UNBALANCED +) (+++++ PARENTHESES ERHUN DETECTION AND MEPORTING. +)
	UTHERWISE: MIXEDTYPE.INTNUMBER := (((INTEGERI/INTEGER2) * INTEGER1)/INTEGER2) * INTEGER1;
3	
	END CASE!
	OPHACTION := OPHACTION := 0; IF UPHACTION > 9 THEN UPRACTION := 0;
	END FUR;
O	ENU KERNEL;
	DEGIN (* PROGRAM EXECUTION *)
	ARTICLINE ('SEGIR SENCHMANK EXECUTION');
5	FUN LUOPCOUNTER 18 8 TO ARRAYSIZE DU
.5	AHNAYZ (LOOPCOUNTER) := LOOPCUUNTEN;
	FUH INBUFFERMTH := 3 TO BUFFERMAX DU INBUFFER(INBUFFERMTH) := 'A';
⁹ ()	INDUFFER[10] := STARTCOUE; INDUFFERPTH := 0;
	FUR TIMINGLUOPER := 3 TO TIMINGCONTROL DO
	AEMNEL;
	WHITELINE (MIXEDTYPE, 'END EXECUTION');
C	END.
0	
	•
Ş	
	·
3	
\cup	
ت	

Tab 4
Interactive "C" Source

റ

O

C

c ()

0

3

ن ر

```
/* PROGRAM DCTBENCHMARK */
               /* Final version, with direct code. At Interactive 270CT80 */
0
              #define ARRAYSIZE 125
              #define OPRACTIONLOOPS 100
              #define IOPORT 32
               #define TESTBYTE 85
               #define INT2 -150
0
               #define MSGLENGTH 80
               #define STARTCODE 'S'
               #define IOLOOPS 575
               #define TIMINGCONTROL 12
               #define NUMBERMSGS 200
               #define INT1 300
C
               #define BITMASK 1
               #define BUFFERMAX 15
               /* Global data */
              int timloop, whilentr, opaction;
              int inbufptr, outbfptr;
3
              int array1[ARRAYSIZE+1],array2[ARRAYSIZE+1];
               int loopcntr, index;
              char inbuffer[BUFFERMAX*1];
              char newchar;
9()
              char notbyte; /* Global for reference in asm .... */
               struct record1 (
                      char charbufr[BUFFERMAX+1];
                      int intnumbr;
               struct record1 mixtype;
               /* End of data declarations */
                              Main Program
               main()
                   printf("Begin benchmark execution\n");
                   for(loopentr = 0; loopentr <= ARRAYSIZE; loopentr++)</pre>
                      array2[loopcntr] = loopcntr;
                   for(inbufptr = 0; inbufptr <= BUFFERMAX; inbufptr++)</pre>
                      inbuffer[inbufptr] = 'A';
                                                                                   B-17
                   inbuffer[10] = STARTCODE;
                   inbufptr = outbfptr = 0;
```

```
for(timloop = 0; timloop <= TIMINGCONTROL; timloop++)
                     kernel();
                  for(index = 0; index <= BUFFERMAX; index++)</pre>
                    printf("%c", mixtype.charbufr[index]);
                  printf("%d\n", mixtype.intnumbr);
                  printf("End execution\n");
             /*Function definitions */
             /* KERNEL: Exercises the support functions. */
             kernel()
                  char outbufr[BUFFERMAX + 1];
()
                  notbyte = ~TESTBYTE;
                  for(loopentr = 0; loopentr <= IOLOOPS; loopentr++)</pre>
                                             _notbyte; /* a <-- byte to be output */
                              asm lda.nn
                                             c, IOPORT; /* c <-- output port */
0
                              asm
                                  ldrn
                                                      /* do the output */
                                   out.cr
                     }
                  for(loopentr = 0; loopentr <= ARRAYSIZE; loopentr++)
                    array1[loopcntr] = array2[ARRAYSIZE - loopcntr];
01
                  for(loopcntr = 0; loopcntr <= NUMBERMSGS; loopcntr++)</pre>
                       do {
                          newchar = getbyte();
                       } while(newchar != STARTCODE);
                       whilentr = 0;
                       while(whilentr++ < MSGLENGTH)
                          if (((newchar = getbyte() ) >> 1 ) & BITMASK )
                             putbyte(newchar,outbufr);
                             putbyte(newchar,outbufr);
                       } /* End while */
                   } /* End for */
                   opaction = 0;
                   for(loopentr = 0; loopentr <= OPRACTIONLOOPS; loopentr++)
                     switch (opaction)
                       case 0:
                       case 3:
                       case 9: mixtype.charbufr[opaction] =
                                              (TESTBYTE >> 3);
                               break:
                                                                                   B-18
                        case 1:
                       case 4:
                        case 7: mixtype.charbufr[opaction] =
```

```
Iftrot(TESTBYTE, 2) & BITMASK;
                                break;
                        case 2:
                        case 6: for(index = 0; index <= BUFFERMAX; index++)</pre>
                                   mixtype.charbufr[index] = inbuffer[index];
                        default: mixtype.intnumbr = ((((INT1/INT2)*
                                    INT1)/INT2)*INT2) * INT1;
                                 break;
                      } /* End switch */
3
                     if (++opaction > 9) opaction = 0;
                   } /* End for */
              } /* End kernel */
              /* GETBYTE: Get a character from inbuffer. Maintain inbufptr. */
()
              char getbyte()
              (
                   char rtnbyte;
                   rtnbyte = inbuffer[inbufptr];
                   if (++inbufptr > BUFFERMAX) inbufptr = 0;
0
                   return(rtnbyte);
              }/* End getbyte */
              /* PUTBYTE: Put a character in putbuffer. Maintain outbfptr. */
              putbyte(inchar, putbuffr)
                   char inchar;
C(
                   char putbuffr[];
              {
                   putbuffr[outbfptr] = inchar;
                   if (++outbfptr > BUFFERMAX) outbfptr = 0;
              } /* End putbyte */
0
              /* LFTROT: Rotate a byte left, by number of bits. */
              char Iftrot(rotbyte, number)
                   char rotbyte, number;
                                   ldr.ixd b,ix+8; /* b <-- number of shifts */</pre>
                      25m
                                   Idr.ixd a,ix+6; /* a <-- byte to be shifted */
                      asm
                                                 /* rotate the byte, while */
                      asm
                           Irot:
                                   ricr a;
                                                 /* decrementing b, jump on not zero */
                                   djnze irot;
                      asm
                                   id.ixdr ix+6,a; /* put rotated byte back */
                      asm
                /* the return will put (ix+6) into the hi register pair */
              return (rotbyte);
              } /*End iftrot */
```

Ü

Tab 5
Whitesmith "C" Source

O

0

C

```
1: ./*
   2:
       #
              DCT BENCHMARK PROGRAM IN WHITESMITH'S C
   3:
       *
              Frank P. MacLachlan
   4:
   5:
              13-0ct-80
   5:
   7:
       */
  1
                             /* file containing standard definitions */
      #include <std.h>
  10:
  11: #define
              ARRAYSIZE
                              125
 12: #define
              BITMASK
                             001
  13: #define
               BUFFERMAX
                              15
  14: #define
               INT1
                              300
                              -150
  15: #define
               INT2
○ 16: #define
               IOLOOPS
                              575
  17: #define
               IOPORT
                              0xff
                                     /* adjust if conflict */
  13: #define
               MSGLENGTH
                             80
  19: #define
               NUMBERMSGS
                              200
  20: #define
               OPRACTIONLOOPS
                             100
  21: #define
               STARTCODE
                              151
O 22: #define
               TESTBYTE
                              35
  23: #define
               TIMINGCONTROL
  24:
  25: /*
  26:
       ¥
              EXTERNAL VARIABLES:
  27:
       ¥
              Due to an anomaly in the Whitesmith C compiler, all
0 28:
       *
              external variables must be given initial values.
  29:
       ¥
              Therefore, external variables which require no initialization
  30:
       *
              must be initialized to satisy the compiler.
  31:
       #/
  32:
              inbuf[BUFFERMAX+1]
      char
                      33:
                        'S', 'A', 'A', 'A', 'A', 'A'};
07
              array1[ARRAYSIZE+1] {0};
  but int
  36: int
              array2[ARRAYSIZE+1] {0};
  37: int
              inbufp {0}:
  38: int
              outbuf# {0};
  39:

    40: struct

  41:
                      charbuf[BUFFERMAX+1];
               char
  42:
               int
                      intnumber;
  43: } mixedtype { "", 0 }:
  44:
  45:
  46:
      /*********************************
  47:
  48:
                      Main Program
  49:
  50:
      51:
  52:
      _main()
<sup>3</sup>53:
  54:
              resister loopctr:
                                      /# for speed #/
  55:
              static int timinglooper;
  56:
  57:
              putfmt("Besin benchmark execution\n");
  58:
              for (loopetr = 0: loopetr <= ARRAYSIZE: ++loopetr)
 3591
                      array2[loopctr] = loopctr;
  6^~
              for (timinslooper = 0; timinslooper <= TIMINGCONTROL; ++timinslooper)
   ٠.,٠
                                                                        B-21
```

ڗ

```
Kernal (7)
 or.
  621
              for (loopetr = 0; loopetr <= BUFFERMAX; ++loopetr)
  63:
                     putfmt("%a", mixedtype.charbuf[loopctr]);
 641
              putfmt("\n%s\nEnd execution\n", mixedtype.intnumber);
  65:
  66:
  57:
  49
      2 70:
                     Support Functions
  71:
  72:
      73:
  74:
              get a character from inbuf, increment inbufp to
  75:
             next character.
0
  76:
      */
  77: char setbyte()
  73:
  79:
             static char c:
  30:
  31:
              c = inbuf[inbufp];
g 32:
              if (++inbufe > BUFFERMAX)
  33:
                     inbufp = O:
  84:
             return (c);
  35:
  36:
  87:
  38: /*
  39:
       *
              kernal is the main function in the benchmark program.
             It is declared VOID to indicate that no return value
  90:
       *
  91:
              is expected. VOID is defined in the file STD.H as
  92:
       *
              being equivalent to int (16 bit integer).
  93:
      #/
0 94
      VOID kernal()
  (٠)
  96:
              static char outbuf[BUFFERMAX+1];
  97:
              static char newchar?
  98:
              static int opaction;
  99:
              resister loopetr, whilectr, i; /* for speed */
C100:
              for (loopetr = 0; loopetr <= IOLOOPS; ++loopetr)
 102:
                     out(IOPORT, ~TESTBYTE); /* standard library output routine */
              for (loopetr = 0; loopetr <= ARRAYSIZE; ++loopetr)
 103:
 104:
                     array1[loopetr] = array2[ARRAYSIZE-loopetr];
 105:
              for (loopetr = 0; loopetr <= NUMBERMSGS; ++loopetr) {
 1061
                     while ((newchar = setbyte()) != STARTCODE)
ો07:
 108:
                     for (whilectr = 0; whilectr < MSGLENGTH; ++whilectr)
 109:
                             if (((newchar = setbyte()) >> 1) & 1)
 110:
                                     putbyte(newchar, outbuf);
 111:
                             else
 112:
                                     Putbyte(newchar, outbuf);
Q13:
             3
 114:
              opaction = O:
 115:
              for (loopetr = 0: loopetr <= OPRACTIONLOOPS: ++loopetr) {
 1168
                     switch(opaction) {
 117:
 118:
                             case OI
319:
                             case 3:
 1204
                             case 91
```

J

```
1216
                                        mixedtype.charburlopactions = (IES(BY(E 32 B)))
 122:
                                        breaki
€123:
 124:
                                case 1:
125:
                                case 4:
                                case 7:
 126:
                                        mixedtype.charbuf[opaction] = 1rot(TESTBYTE, 2)
 127:
 1.
                                        break;
J129':
 130:
                                case 2:
 131:
                                case 6:
                                        for (i = 0; i <= BUFFERMAX; ++i)
 132:
 133:
                                                 mixedtype.charbuf[i] = inbuf[i];
 134:
                                        break;
್ಷ135៖
 136:
                                default:
 137:
                                        mixedtype.intnumber = ((((INT1/INT2) * INT1)
 138:
                                           / INT2) * INT2) + INT1;
 139:
                       if (++opaction > 9)
 140:
0141:
                                opaction = O;
 143:
               } /** end kerne1 **/
 144:
 145:
146: /*
               Put a byte into the next position in Putbuf
 _147: */
"148: VOID putbyte(inchar, putbuf)
 149:
               char
                       inchar;
- 150:
                       Putbuf[];
               char
 151:
 152:
oi53;
               putbufCoutbufpl = inchart
               if (++outbufp > BUFFERMAX)
 155:
                       outbuf# = 0;
 156:
 157:
 158:
 159: /*
               rotate byte n left b bit positions
C160:
               (8 bit rotate)
       */
 161:
 162% char Irot(n. b)
 163:
               char
                       n:
 164:
               char
 165:
 366:
 1678
               b &= 007;
                                        /# O..7 #/
               return((n << b) : (n >> 8-b));
. 168:
 1691
```

C

0

```
C
       PROGRAM DCT HOL BENCHMARK IN MICROSOFT FORTRAN-80 FOR CP/M
C
       WRITTEN BY: CAPT B.F. BRADY, U.S.M.C
C***** VARIABLE DECLARATIONS FOR MAIN PROGRAM *****
      IMPLICIT LOGICAL(L-R)
C
      INTEGER OTBFPT, INBFPT, TIMLPR, INTNUM, ARRAY2
      LOGICAL*1 CHRBUF, INBUFF, BITMSK, TSTBYT
C**** DEFINE GLOBALS IN COMMON :
      COMMON //OTBFFT, INBUFF(16), INBFPT, CHRBUF(16), ARRAY2(126),
               INTNUM, BITMSK, TSTBYT
C**** PRESET DATA :
      DATA INBUFF/10*1HA,1HS,5*1HA/,BITMSK/1/,OTBFPT/O/,TSTBYT/1HU/
C**** BENCHMARK MAIN PROGRAM
C**** WRITE OUT THE START MESSAGE
      WRITE(5,1)
C**** INITIALIZE ARRAY2
      DO 102 TIMLPR=1,126
        ARRAY2(TIMLPR)=TIMLPR
C**** END INITIALIZE LOOP
      CONTINUE
102
      DO 105 TIMLPR=0,12
C**** BEGIN MAIN TIMING LOOP
        CALL KERNEL
C**** END MAIN TIMING LOOP
105
     CONTINUE
C**** WRITE OUT THE END MESSAGE
       WRITE(5,2)
C**** END OF MAIN PROGRAM
C**** FORMAT STATEMENTS :
      FORMAT( ' Besin BENCHMARK Execution. ')
      FORMAT( ' End BENCHMARK Execution. ')
      STOP
      END
C**** SUBROUTINES AND FUNCTIONS :
      FUNCTION GETBYT(DUMARG)
      LOGICAL*1 INBUFF, DUMARG
C**** DUMARG IS USED ONLY BECAUSE A PARAMETER IS
C**** REQUIRED FOR A FUNCTION CALL
      INTEGER OTBFPT, INBFPT, LOCPNT
      COMMON //OTBFPT, INBUFF(16), INBFPT
      LOCPNT=INBFPT
```

Ü

```
0
                INBFPT=INBFPT+1
                IF (INBFPT.GT.16) INBFPT=1
                GETBYT= [NBUFF (LOCPNT)
                RETURN
         C**** END GETBYT
                END
         10
                SUBROUTINE PUTBYT (INCHAR, PUTBUF)
         C
                LOGICAL*1 INCHAR, PUTBUF(16)
         C
                INTEGER OTBFFT
         C
                COMMON //OTBFPT
         C
                PUTBUF(OTBFPT)=INCHAR
                OTBFPT=OTBFPT+1
                IF(OTBFPT.GT.16) CTBFPT=1
                RETURN
         C**** END PUTBYT
                END
         C
                SUBROUTINE KERNEL
0
                IMPLICIT LOGICAL(L-R)
         C
                LOGICAL*1 NEWCHR, INBUFF, TSTBYT, BITMSK, CHRBUF, OTBUFF (14), START,
                           BYTE
         C
                INTEGER OTBFPT, INBFPT, INTNUM, LPCNTR, TIMLPR, WHLCNT, CPRACT, ARG2,
\mathcal{I}()
                         ARRAY2, ARRAY1 (126), INTOR1, INTOR2, TEMP
         C
                COMMON //OTBFPT, INBUFF(16), INBFPT, CHRBUF(16), ARRAY2(126),
                          INTNUM, BITMSK, TSTBYT
         C
                DATA START/1HS/, INTGR1/300/, INTGR2/-150/
         C
                DO 112 LPCNTR=0,575
                   BYTE=.NOT.TSTBYT
                   CALL OUT(BYTE, 200)
                CONTINUE
          112
         C
                DO 115 LPCNTR=1,126
                   ARG2=127-LPCNTR
                   ARRAY1 (LPCNTR) = ARRAY2 (ARG2)
                CONTINUE
          115
         C
                DO 127 LPCNTR=0,200
         C**** BEGIN MESSAGE LOOP
          119
                  CONTINUE
                  IF (GETBYT (BYTE) . NE. START) GOTO 119
                  WHLCNT=0
                IF(WHLCNT.GT.80)GOTO 126
          122
3
         C**** BEGIN WHILE LOOP
                    NEWCHR=GETBYT(BYTE)
                    TEMP=NEWCHR
```

```
IF(RTSHFT(TEMP,1).AND.BITMSK)GOTO 124
                     CALL PUTBYT (NEWCHR, OTBUFF)
                     GOTO 125
         124
                 CONTINUE
                     CALL PUTBYT (NEWCHR, OTBUFF)
         125
                 CONTINUE
                   WHLCNT=WHLCNT+1
         C**** END WHILE LOOP
                   GOTO 122
         126
               CONTINUE
         C**** END MESSAGE LOOP
0
               CONTINUE
               OPRACT=0
               DO 138 LPCNTR=0.100
         C**** BEGIN OPERATOR ACTION LOOP
                  INDEX=OPRACT+1
                  GOTO(129,131,128,129,134,131,129,134,128),OPRACT
         C**** CASE OF 0,3 OR 9
         123
                   CHRBUF(INDEX)=RTSHFT(TSTBYT,3)
         C**** END 0,3 OR 9
                   GOTO 135
         C**** CASE OF 1,4 OR 7
         129
                   CHRBUF(INDEX)=LFTROL(TSTBYT,2).AND.BITMSK
         C**** END 1,4 OR 7
                   60T0 135
         C**** CASE OF 2 OR 6
         131
                   DO 133 ARG2=1,16
                      CHRBUF(ARG2)=INBUFF(ARG2)
         133
                   CONTINUE
         C**** END 2 OR 6
                   COTO 135
         C**** OTHERWISE :
                   INTNUM=((((INTGR1/INTGR2)*INTGR1)/INTGR2)*INTGR2)+INTGR1
         C**** END OTHERWISE
         C**** END OF CASE SIMULATION
         135
                  CONTINUE
                  OPRACT≈OPRACT+1
                  IF(OPRACT.GT.9) OPRACT=0
               CONTINUE
         138
         C**** END OPERATOR ACTION LOOP
         C**** END KERNEL
               RETURN
               END
```

う

Tab 7
Cromemco RATFOR Source

Ċ

```
#PROGRAM DCT HOL BENCHMARK in RATFOR for CP/M
#Written by: Capt B.F. BRADY, U.S.M.C
# INCLUDE REGLELS.RAT # need this for TSW RatFor only
#DEFINE Constants :
DEFINE(arsize, 126)
DEFINE(inloop, 575)
DEFINE(opacle, 100)
DEFINE(timet1,12)
DEFINE(ioport, 200)
DEFINE(nms95,200)
DEFINE(tstbyt,85)
DEFINE(inter1,300)
DEFINE(inter2, (-150))
DEFINE(msslen,80)
DEFINE(strtcd, 'S')
DEFINE(bufmax, 16)
#Variable declarations for main program
INTEGER otbfpt, inbfpt, timlpr, intnum, array2
LOGICAL*1 chrbuf, inbuff, bitmsk, temp
#DEFINE slobals in COMMON :
COMMON //otbfpt,inbuff(bufmax),inbfpt,chrbuf(bufmax),
         array2(arsize), temp, intnum, bitmsk
#PRESET DATA:
DATA inbuff/10#'A','S',5#'A'/,bitmsk/1/,otbfpt/0/
#Benchmark Main Program
   WRITE(5.1) #put start message on console
   # initialize array2 elements to equal their index
   FOR(inbfpt=1;inbfpt<=arsize;inbfpt=inbfpt+1)
       array2(inbfpt) = inbfpt
   # end initialize loop
   inbfpt = 0
   FOR(tim)pr=1;tim)pr<=timct);tim)pr=tim)pr+1)
   (# BEGIN main timing loop
      CALL kernel
   3# END main timing loop
   WRITE(5,2)chrbuf,inthùm
   # put out record values and end message on console
#FORMAT Statements:
 1 format(' besin benchmark execution')
 2 format(1x,16A1,I4,' end execution')
STOP
END
#end of Main Program
```

 \circ

```
#Subroutines and Functions :
3
       FUNCTION setbyt(dumars)
       LOGICAL*1 inbuff, dumars
                                  #dumars is used only because a parameter is
                                  #required for a FUNCTION call
        INTEGER
                  otbfet, inbfet, lacent
0
       COMMON //otbfpt,inbuff(bufmax),inbfpt
           locpnt = inbfpt
           inbfpt = inbfpt + 1
           IF(inbfrt .GT. bufmax)
              inbfpt = 1
10
           setbyt = inbuff(locent)
       RETURN
       END
        #end setbyt
· 0
        SUBROUTINE putbyt(inchar, putbuf)
        LOGICAL*1 inchar, putbuf(bufmax)
        INTEGER otbfpt
COMMON //otbfpt
           putbuf(otbfpt) = inchar
           otbfpt = otbfpt + 1
           IF(otbfmt .GT. bu/max)
              otbfpt = 1
       RETURN
       END
        #end putbyt
        SUBROUTINE kernel
\mathbf{C}
       LOGICAL*1 newchr, inbuff, temp, bitmsk, chrbuf, otbuff(bufmax), start
        INTEGER
                  otbfpt,inbfpt,intnum,timlpr,lpcntr,whlcnt,opract,arg2,
                  array2, array1(arsize)
        COMMON //otbfpt,inbuff(bufmax),inbfpt,chrbuf(bufmax),
.0
                 array2(arsize), temp, intnum, bitmsk
       DATA
                 start/strtcd/
            FOR(|pentr=1;|pentr<=ioloop;|pentr=|pentr+1)
0
               {#BEGIN I/O Loop
                temp = (.NOT.tstbyt)
                CALL out(temp, ioport)
               SHEND I/O Loop
            FOR(lpcntr=1;lpcntr<=arsize;lpcntr=1pcntr+1)
               CABEGIN Array Loop
 ٣
                ars2 = (arsize+1) - lpcntr
                array1(lpcntr) = array2(arg2)
```

Ü

```
}#END Array Loop
0.7
           FOR(lpcntr=1;lpcntr<=nmsss;lpcntr=1pcntr+1)
             {#BEGIN Message Loop
              REPEAT
                                       #use temp as a dummy here since
                 newchr = setbyt(temp)
                                       #at least one argument is needed for ca
              UNTIL(newchr .EQ. start)
              whicht = 0
              WHILE(whicht < msslen)
                 {#BEGIN While Loop
                  newchr = setbyt(temp)
                  temp = newchr
                  CALL rtshft(temp,1)
0
                  IF(temp .AND. bitmsk)
                     CALL putbyt(newchr, otbuff)
                  ELSE
                     CALL putbyt(newchr, otbuff)
                  whicht = whicht + 1
}#END While Loop
             }#END Message Loop
           opract = 0
           FOR(lpcntr=1;lpcntr<=opaclp;lpcntr=1pcntr+1)
              {#BEGIN Operator Action Loop
               3
              ## simulate CASE statement with IF-ELSE-IF chain
              IF((opract.EQ.O).OR.(opract.EQ.3).OR.(opract.EQ.9))
                 {#CASE of 0,3 or 9
                  temp = tstbyt
                  CALL rtshft(temp,3)
00
                  chrbuf(opract) = temp
                 }#End 0,3 or 9
              ELSE IF((opract.EQ.1).OR.(opract.EQ.4).OR.(opract.EQ.7))
                 {#CASE of 1,4 or 7
                  temp = tstbyt
 O
                  CALL Iftro1(temp,2)
                  temp = (temp .AND. bitmsk)
                  chrbuf(opract) = temp
                 3#End 1,4 or 7
               ELSE IF((opract.EQ.2).OR.(opract.EQ.6))
 O
                 {#CASE of 2 or 6
                  FOR(arg2=1;arg2<=bufmax;arg2=arg2+1)
                     chrbuf(ars2) = inbuff(ars2)
                 }#End 2 or 6
              ELSE
                 {#OTHERWISE :
                  intnum=(((((intsr1/intsr2)#intsr1)/intsr2)#intsr2)+intsr1
                 }#End Otherwise
               #End of CASE Simulation
               opract = opract + 1
               IF(opract .GT. 9)
                 opract = 0
```

>#END Operator Action Loop
#end kernel
RETURN
END

0

()

0

O

0

0

10

Tab 8
PASCAL/MT Source

0

.)

O

```
0
        (*$L+*)
        (* DCT Benchmark Program in Pascal/MT for CP/M *)
             written by : Capt B.F. BRADY, U.S.M.C.
        (*
        PROGRAM DCTBENCHMARK;
        CONST
           arraysize = 125;
0
           numbermses = 200;
           opracloops = 100;
           ioport
                      = 200;
                      = 575;
           ioloops
           timinscntrl= 12;
                      = 85;
           testbyte
           intemer1
                      = 300;
                      =-150;
           integer2
           bitmask
                          1;
           msslensth
                         801
           startcode
                      = 'S':
           startbyte
                     = 101
3
           buffermax =
                         15;
        TYPE
           buffertype = PACKED ARRAY[O..buffermax] of CHAR;
           mixedtype = RECORD
              charbuffer : buffertype:
0
              intnumber : INTEGER
           END (*RECORD mixedtype*);
        VAR
           timinglooper, inbufptr, outbufptr : INTEGER;
0
           array1, array2 : ARRAY[0..arraysize] of INTEGER;
           inbuffer : buffertype:
           mixrec : mixedtype;
        FUNCTION setbyte : CHAR;
0
           BEGIN
              setbyte := inbuffer[inbufptr];
              inbufetr := inbufetr + 1;
              IF inbufetr > buffermax THEN inbufetr := 0
           END (*setbyte*);
` (3
        PROCEDURE putbyte (VAR inchar : CHAR; VAR putbuffer : buffertype);
           BEGIN
              putbufferCoutbufptr] := inchar; -
              outbufetr := outbufetr + 1;
              IF outbuffer > buffermax THEN outbuffer := 0
           END (*Putbyte*);
0
        PROCEDURE LFTROL(VAR temp : INTEGER; bits :INTEGER);
        BEGIN
           INLINE(
                          "LDA / bits/
                          "MOV B,A /
0
                          "LDA / temp/
                  [LROT]/ "RLC
```

```
"DCR B /
O \cdot
                           "JNZ / LROT/
                           "STA / temp);
        END:
        PROCEDURE kernel:
           VAR outbuffer : buffertype:
                loopcounter, whilecounter, opraction, temp : INTEGER;
                newchar : CHAR;
           BEGIN
\mathbf{C}
              FOR loopcounter := 0 TO icloops DO
                   BEGIN
                      temp := testbyte;
                      (* use in-line direct code to complement temp and put *)
                      (* it out the specified port *)
                      INLINE( "LDA /temp/
                              "CMA /
                              "OUT /ioport);
              END
                   (*FOR*);
              FOR loopcounter := 0 TO arraysize DO
C
                  array1[loopcounter] := array2[arraysize - loopcounter];
              FOR loopcounter := 0 TO numberms#s DO
                  BEGIN
                     REPEAT
                        newchar := setbyte
CT
                     UNTIL newchar = startcode;
                     whilecounter := O;
                     WHILE whilecounter C msslensth DO
                        BEGIN
                           newchar := setbyte;
                           temp := SHR(newchar,1);
                          IF (ODD(temp) AND ODD(bitmask)) THEN
                              putbyte(newchar,outbuffer)
                           ELSE
                              putbyte(newchar, outbuffer);
 C
                           whilecounter := whilecounter + 1;
                        END
                              (* WHILE*);
                     END
                         (*FOR*);
           opraction := 0:
           FOR loopcounter := 0 TO opracloops DO
 C
              BEGIN
                  temp := testbyte;
                  CASE opraction OF
                           # mixrec.charbuffer[opraction] := CHR(SHR(temp,3));
                     0,3,9
 0
                     1,4,7
                            : BEGIN
                                 LFTROL(temp, 2);
```

O

```
mixrec.charbuffer[opraction] := CHR(ODD(temp) AND ODD(bitmask))
```

END:

2.6 | mixrec.charbuffer := inbuffer;

ELSE

mixrec.intnumber := ((((inteser1 DIV inteser2)*inteser1)
DIV inteser2)*inteser2) + inteser1

END (*CASE*);

opraction := opraction + 1;
IF opraction > 9 then opraction := 0

END (*FOR*); END (*kernel*);

BEGIN (* MAIN PROGRAM EXECUTION *)

WRITELN(' Besin Benchmark Execution');

FOR timinslooper := 0 TO arraysize DO array2[timinslooper] := timinslooper;

FOR timinglooper := 0 TO buffermax DO inbuffer[timinglooper] := 'A';

inbuffer[startbyte] := startcode; inbuffer := 0; outbuffer := 0; FOR timinglooper := 0 TO timingcntr1 DO kernel; writeln(mixrec.charbuffer,mixrec.intnumber); WRITELN(' End Benchmark Execution')

END.

0

0

O

0

()

0

C

0

 \odot

Tab 8
PASCAL/MT Source

ા

Э

```
0.
        (#$I
              intersperse pascal source in asmbl source
        (* DCT HOL BENCHMARK in Pascal/Z from Ithaca Intersystems *)
        (* Written by: Capt B.F. BRADY, U.S.M.C. *)
        PROGRAM DCTBENCHMARK;
        CONST
           arraysize = 125;
           numbermsss = 200;
           opracloops = 100;
           ioport
                       = 200;
· O
           ioloops
                       = 575;
           timinscntrl=
                          2;
           testbyte
                         35;
                       = 300;
           integer1
           integer2
                       =-150;
           bitmask
                       #
                           1;
           msslensth
                          80;
           startbyte
                      =
                         10;
           buffermax = 15
        TYPE
           buffertype = ARRAY[O..buffermax] of CHAR;
: o
           mixedtype = RECORD
              charbuffer : buffertype;
              intnumber : INTEGER
           END (*RECORD mixedtype*);
           timinglooper : INTEGER;
           inbufetr, outbufetr : INTEGER;
           array1, array2 : ARRAY[O..arraysize] of INTEGER;
           inbuffer : buffertype:
           mixrec : mixedtype:
 \bigcirc
        FUNCTION setbyte : CHAR;
           VAR
              localent : INTEGER:
: 0
           BEGIN
              localent := inbufetr;
              inbufetr := inbufetr + 1;
              IF inbufptr > buffermax THEN inbufptr := 0;
              setbyte := inbuffer[localpnt]
           END (*setbyte*);
0
        PROCEDURE putbyte (VAR inchar : CHAR; VAR putbuffer : buffertype);
           BEGIN
              PutbufferCoutbufptr] := inchar;
              outbufptr := outbufptr + 1;
0
              IF outbufetr > buffermax THEN outbufetr := 0
           END (*Putbyte*);
```

```
PROCEDURE ANDCHR(VAR temp : INTEGER; mask : INTEGER); EXTERNAL;
       PROCEDURE OUTPUT (VAR temp : INTEGER; port : INTEGER); EXTERNAL;
       PROCEDURE LFTROL(VAR temp : INTEGER; bits : INTEGER); EXTERNAL;
       PROCEDURE RTSHFT(VAR temp : INTEGER; bits : INTEGER); EXTERNAL;
       PROCEDURE kernel:
          VAR
Э
             outbuffer : buffertype:
             loopcounter, whilecounter : INTEGER;
             opraction, temp : INTEGER:
             newchar : CHAR;
          BEGIN
7
       writeln('kernel');
       writelm('ioloops');
             FOR loopcounter := 0 TO isloops DO
                BEGIN
                   temp := testbyte;
                   DUTPUT(temp,ioport)
                END (*FOR*);
       writeln('arraylooms');
             FOR loopcounter := 0 TO arraysize DO
                array1[loopcounter] := array2[arraysize - loopcounter];
90
       writeln('mssloops');
             FOR loopcounter := 0 TO numbermses DO
                BEGIN
                   REPEAT
                      newchar := setbyte
                   UNTIL newchar = 'S';
                   whilecounter := O:
                   WHILE whilecounter < msslensth DO
                      BEGIN
                         newchar := setbyte;
0
                         temp := ord(newchar);
                         RTSHFT(temp,1);
                         ANDCHR(temp,bitmask);
                         IF temp = 1 THEN
                             putbyte(newchar,outbuffer)
                             putbyte(newchar,outbuffer);
                         whilecounter := whilecounter + 1
                            (* WHILE*)
                      END
                   END
                        ( #FOR#) ;
          opraction := O:
       writeln('oprloops');
          FOR loopcounter := 0 TO opracloops DO
```

```
BEGIN
         temp := testbyte;
(*$J9 Compiler option to create CASE Jump Table for 0..9 *)
         CASE opraction OF
            0,3,9 : BEGIN
                        RTSHFT(temp,3);
                        mixrec.charbuffer[opraction] := chr(temp)
                     END:
            1,4,7 : BEGIN
                        LFTROL(temp,2);
                        ANDCHR(temp,bitmask);
                        mixrec.charbuffer[opraction] := chr(temp)
                     END;
            2,6
                   : mixrec.charbuffer := inbuffer;
         ELSE:
            mixrec.intnumber := ((((inteser1 DIV inteser2)*inteser1)
                                    DIV integer2)*integer2) + integer1
         END
               (*CASE*);
         opraction := opraction + 1;
         IF opraction > 9 then opraction := 0
      END
          (*FOR*)
   END
       (*kernel*);
BEGIN (* MAIN PROGRAM EXECUTION *)
   WRITELN( 'Besin Benchmark Execution');
   FOR timinslooper := 0 TO arraysize BO
       array2[timinslooper] := timinslooper;
   FOR timinglooper := 0 TO buffermax DO
       inbuffer[timinglooper] := 'A';
   inbuffer[startbyte] := 'S';
   inbufetr := Of
   outbufptr := O;
   FOR timinglooper := 0 TO timingentr1 DO
   WRITELN(mixrec.charbuffer,mixrec.intnumber);
   WRITELN(chr(7), ' End Benchmark Execution') .
END.
```

Tab 10 PLI-80 Source (Version 1)

 O^{-1}

```
/* BENCHMARK PROGRAM IN PLI/80 FOR CP/M */
       /* Written by: Capt B.F. BRADY, U.S.M.C #/
       BENCHMARK_PLI_80:
         PROCEDURE OPTIONS (MAIN) :
           "REPLACE /* DEFINE CONSTANTS */
              opr_action_loops BY
                                    100.
              timing_control
                                BY
                                      1,
              array_size
                                BY
                                    125.
                                BY
              io_loops
                                    575,
                                BY
              io_port
                                    200,
                                BY
                                    200,
              number_msss
0
              test_byte
                                BY
                                  1551B4,
                                BY
              buffer_max
                                     15,
                                BY
                                    300.
              integer1
                                BY
              integer2
                                    150.
              bit_mask
                                BY
                                   /1/B.
              mss_lensth
                                BY
                                     80,
              start_byte
                                BY
                                     10,
                                BY
              start_code
                                    'S';
       DEFINE EXTERNAL PROCEDURE ENTRIES
       */
         DCL
\supset
           OUTPUT
                      ENTRY(BIT(8),BIT(8)),
           RTSHFT
                      ENTRY(BIT(8), FIXED(7)),
           LFTROT
                      ENTRY(BIT(8),FIXED(7));
       DEFINE VARIABLES FOR MAIN PROGRAM :
0.
       */
         DCL
             mixed_type STATIC,
              2 character_buffer(0:buffer_max) CHAR(1),
              2 integer_number FIXED BINARY;
         DCL
           input_buffer(0:buffer_max) CHAR(1) INITIAL((10)'A','S',(5)'A') STATIC,
           out_buffer_pointer FIXED BINARY(7) STATIC,
           in_buffer_pointer FIXED BINARY(7) STATIC,
           timins_loop_counter FIXED BINARY STATIC,
           i FIXED BINARY(7) STATIC,
           array_2(array_size) FIXED BINARY STATIC:
       /#
       PROCEDURE AND FUNCTION DEFINITIONS:
         PUT_BYTE:
           PROCEDURE(in_character.put_buffer):
           DCL
              in_character CHAR(1).
              put_buffer(0:buffer_max) CHAR(1);
           BEGIN:
              Put_buffer(out_buffer_pointer) = in_character;
```

```
0.
             out_buffer_pointer = out_buffer_pointer + 1;
             IF out_buffer_pointer > buffer_max THEN
               out_buffer_pointer = 0;
          END;
        END PUT_BYTE;
        GET_BYTE:
          PROCEDURE RETURNS(CHAR(1)): /* A FUNCTION DEFINITION */
             local_pointer FIXED BINARY(7);
          BEGIN;
             /* SAVE INDEX IN local_pointer FOR THE RETURN */
0
             local_pointer =in_buffer_pointer;
             in_buffer_pointer = in_buffer_pointer + 1;
             IF in_buffer_pointer > buffer_max THEN
               in_buffer_pointer = O;
          END:
          RETURN(input_buffer(local_pointer));
        END GET_BYTE; /* END FUNCTION GET_BYTE */
      KERNEL:
        PROCEDURE:
          DCL
             out_buffer(0:buffer_max) CHAR(1) STATIC,
0
             loop_counter FIXED BINARY STATIC,
             array_1(array_size) FIXED BINARY STATIC,
             new_character CHAR(1) STATIC,
             temp FIXED BINARY(7) STATIC,
             operator_action FIXED BINARY(7) STATIC,
             while_counter FIXED BINARY(7) STATIC:
0/
          BEGIN:
             DO loop_counter = 0 TO io_loops:
              CALL output(io_port,^test_byte);
             END: /* I/O Loop */
             DO loop_counter = 0 TO array_size:
               array_1(loop_counter)=array_2(array_size - loop_counter);
            END: /* Array loop */
            DO loop_counter = 0 TO number_msss:
              DO WHILE(set_byte() ^= start_code);
              END: /* DO WHILE */
              while_counter = O:
\circ
              DO WHILE(while_counter < mss_lensth);
                new_character = set_byte();
                 temp = UNSPEC(new_character);
                CALL rtshft(1,temp);
                 IF (UNSPEC(temp) & bit_mask) THEN
                  CALL put_byte(new_character,out_buffer);
                ELSE
                   CALL put_byte(new_character,out_buffer);
                while_counter = while_counter + 1;
              END: /* WHILE Loop #/
                  /# Message Loop #/
             operator_action = Of
             DO loop_counter = 0 TO opr_action_loops:
         put skip list(operator_action.mixed_type.character_buffer(operator_actior
```

 \Box

```
GOTO CASE(operator_action);
· ()
       CASE(0): ;
       CASE(3): ;
       CASE(9):
                 temp = test_byte;
                 CALL rtshft(3, temp);
0
                 UNSPEC(mixed_type.character_buffer(operator_action)) = UNSPEC(te
                 GOTO END_CASE;
       CASE(1): ;
       CASE(4): ;
       CASE(7):
0
                 temp = test_byte;
                 CALL 1ftrot(2, temp);
                 UNSPEC(mixed_type.character_buffer(operator_action)) =
                 (UNSPEC(temp) & bit_mask);
                 GOTO END_CASE;
       CASE(2): ;
       CASE(6):
                 mixed_type.character_buffer = input_buffer:
                 GOTO END_CASE;
10
       /* OTHERWISE */
       CASE(5): ;
       CASE(8):
                 mixed_type.integer_number =
              ((((inteser1/(-inteser2))*inteser1)/(-inteser2))*(-inteser2))+intese
       END_CASE:
         Put skip list(operator_action,mixed_type.character_buffer(operator_action
                 operator_action = operator_action + 1;
                 IF operator_action > 9 THEN operator_action = 0;
            END: /* Operator Action Loop */
          END:
       END KERNEL;
       /***********************
       /* Besin main prosram execution #/
       /*********************
BEGIN:
           PUT SKIP LIST(' Besin Benchmark Execution');
           input_buffer(start_byte) = start_code;
           DO timing_loop_counter = 0 TO array_size:
             array_2(timins_loop_counter) = timins_loop_counter;
           END: /* initialize array_2 #/
           DO timing_loop_counter = 0 TO timing_control;
             CALL kernels
           END: /* main timing loop */
           PUT SKIP:
           DO i=1 TO buffer_max!
           PUT LIST(mixed_type.character_buffer(i));
           END;
```

PUT LIST(mixed_type.inteser_number);
PUT SKIP LIST(' End Execution');
END;
END BENCHMARK_PLI_80; /* Main Program */

 $O\cdot\cdot$

Ool

[0

. 3

0

O

3

0

1961年 李蒙古

Tab 11
PLI-80 Source (Version 2)

()

Ö

: 0

0

Ü

B-46

```
BMARK: PROCEDURE OPTIONS (MAIN):
0
          DCT BENCHMARK PROGRAM IN PL/I-80
                   WRITTEN BY:
                                  F. P. MACLACHLAN
                                  14-0CT-82
                   :ITAC
0
          ZREPLACE
                                     BY 125,
BY 1 B,
                 ARRAY SIZE
()
                 BITMAŠK
                                      BY 15,
                 BUFFER MAX
                                      BY 300.
                 INTEGER1
                                      BY (-150),
                                                        WON'T ACCEPT (-150)!!
                 INTEGERS
                 IO_LOOPS
IO_PORT
                                      BY 575,
                                      BY 200.
                                                        /# ADJUST #/
0
                                      BY 80.
                 MSG_LENGTH
                 NUMBER_MSGS
                                      BY 200.
                 CPR_ACTION_LOOPS
                                      2Y 130,
                 START CODE
TEST_BYTE
                                      3Y 'S'
                                         '55 B4.
                                                        /* 85 DECIMAL */
                                      BY
 O <
                 TIMING CONTROL
                                      BY 12:
             DCL
                 SHIFTR
                                      ENTRY (BIT(8), FIXED(7)) RETURNS (BIT(8)).
                                      ENTRY (BIT(8), FIXED(?)) RETURNS (BIT(8)),
ENTRY (FIXED(?), BIT(8));
                 ROTATL
                 OUTPUT
             DCL
                                      CHAR,
                 NEW_CHAR
                 OUT_BUFFER(2:15)
                                      CHAR STATIC.
                                                        /* NOTE: COULD USE CHAR(16) */
                 ARRAY1 (Ø:ARRAY_SIZE) FIXED BINARY,
ARRAY2 (Ø:ARRAY_SIZE) FIXED BINARY,
IN_BUFFER (Ø:15) CHAR STATIC INITIAL
                     ( A , A , A ,
                 IN BUFFER PTR
                                      FIXED BINARY STATIC INITIAL (0).
                 LOOP COUNTER
OPR ACTION
OUT BUFFER PTR
TIMING LOOPER
WHILE COUNTER
                                      FIXED
                                            BINARY.
                                      FIXED BINARY.
                                      FIXED BINARY STATIC INITIAL (0).
                                      FIXED BINARY.
                                      FIXED BINARY.
                 1 MIXED_TYPE STATIC.
 0
                    2 CHAR BUFFER (3:15)
2 INT_NUMBER
                                               CHAR.
                                               PIXED BINARY;
                                                                                 B-47
```

17-NOV-82

0

FILENAME: BMPLIS.PLI

GET_BYTE:
PROCEDURE RETURNS (CHAR);

```
FILENAME: BMPLIS.PLI
                                     17-NOV-82
    DCL
         C CHAR:
    C = IN_BUFFER(IN_BUFFER_PTR);
IN_BUFFER_PTR = IN_BUFFER_PTR + 1;
    IF IN BUFFER PTR > BUFFER MAX THEN
         IN_BUFFER_PTR = 0;
    RETURN (C);
    END GET BYTE:
PUT BYTE:
    PROCEDURE (IN_CHAR, PUT_BUFFER);
    DCL
         IN_CHAR CHAR.
         PUT_BUFFER(J:BUFFER_MAX) CHAR;
    PUT_BUFFER(OUT_BUFFER_PTR) = IN_CHAR;
OUT_BUFFER_PTR = OUT_BUFFER_PTR + 1;
IF OUT_BUFFER_PTR > BUFFER_MAX_THEN
         OUT_BUFFER PTR = 0;
    END PUT BYTE;
KERNAL:
    PROC:
    DCL
         OUT_BUFFER(2:BUFFER_MAX) CHAR;
    DC LOOP_COUNTER = Ø TO IO_LOOPS;
CALL OUTPUT(IO_PORT, TEST_BYTE);
         END;
    DO LOOP_COUNTER = Ø TO ARRAY_SIZE;
         ARRAY1(LOOP COUNTER) = ARRAY2(ARRAY SIZE-LOOP COUNTER);
    DO LOOP_COUNTER = Ø TC NUMBER_MSGS;

NEW_CHAR = GET_BYTE();

DO WHILE (NEW_CHAR = START_CODE);
             NEW_CHAR = GET_BYTE();
             END;
         WHILE_COUNTER = 0;
        DO WHILE (WHILE COUNTER < MSG_LENGTH);

NEW_CHAR = GET_BYTE();

IF SHIFTR(UNSPEC(NEW_CHAR), 1) & '1'B THEN

CALL PUT_BYTE(NEW_CHAR, OUT_BUFFER);
                  CALL PUT_BYTE(NEW_CHAR, OUT_BUFFER);
         WHILE COUNTER = WHILE COUNTER + 1;
END; 7* DO WHILE */
END; /* DO LOOP COUNTER */
    OPR_ACTION = 0;
DO LOOP_COUNTER = 0 TO OPR_ACTION_LOOPS;
                                                                                           R-48
```

GOTO CASE(OPR_ACTION);

0

0

3

Ö

0

 \circ

•3

: 0

()

```
FILENAME: BMPLIS.PLI
                                   17-NOV-83
    CASE(0):;
CASE(0):;
     CASE(9):;
                UNSPEC(MIXED_TYPE.CHAR_BUFFER(OPR_ACTION)) = SHIFTR(TEST_BYTE, 3);
                GOTO END_CASE;
     CASE(1):;
     CASE(4):;
     CASE(7):;
                UNSPEC(MIXED_TYPE.CHAR_BUFFER(OPR_ACTION)) =
                   ROTATL(TEST_BYTE, 2) & BITMASK;
                GOTO END_CASE;
7
     CASE(2):;
     CASE(6):;
                MIXED_TYPE.CHAR_BUFFER = IN_BUFFER;
                GOTO END_CASE;
0
     CASE(5):;
     CASE(8):;
                MIXED_TYPE.INT_NUMBER = ((((INTEGER1 / (-150)) * INTEGER1) / (-150)) * (-150)) + INTEGER1;
     END_CASE:;
            OPR ACTION = OPR ACTION + 1;
IF OPR_ACTION > 3 THEN
0
                OPR_ACTION = 0;
             END; /* DO LCOP_COUNTER = Ø TO OPR_ACTION_LOOPS */
         END KERNAL;
\cdot
      /本本本本本本本本本本本本本本本本本本本本本本本本本本本本本本本本本本本本
          START OF MAIN PROGRAM
         DCL
             I FIXED BINARY;
∙⊃
         PUT SKIP LIST ('BEGIN BENCHMARK EXECUTION');
DO I = 0 TO BUFFER_MAX;
            ARRAY2(I) = I;
             END:
         DO TIMING_LOOPER = 0 TO TIMING_CONTROL;
            CALL KERNAL();
             END:
         DO I = 2 TO 15;
             PUT LIST (MIXED_TYPE.CHAR_BUFFER(I));
             END;
         PUT LIST
             (MIXED_TYPE.INT NUMBER, 'END EXECUTION');
         IND BMARK:
```

C

Tab 12 PLMX Source

, O

0

 \circ O

0

R_SA

```
.Zanchmera wourea
0
     PLAX COMPILER VERSION 2.4
     COPYRIGHT (C) 1980, SYSTEMS CONSULTANTS, INC.
     END OF FAST COMPILATION
     DUO ERROR(S) DETECTED
             DOT BENCHMARK - PLIM VERSION
     ्
     bmark: do;
        declare
                 literally
                                     'literally',
           equ
                                     1261
                                            /* O. . 125 */
           ARRAY$SIZE
                             equ
                                     'O1h'.
           BITMASK
                             equ
           BUFFER$SIZE -
                                     7167,
                                            /* 0...15 */
                             equ
                                     'ODh'
           CR
                             equ
           INT1
                                     '300 ',
                             equ
           INT2
                                     7-1507
                             equ
           IO*LOOPS
                                     15751
                             equ
           IO$PORT
                                     .500.
                                             /* adjust */
                             equ
                                     OAh "
           LF.
                             equ
                                     1801,
           MSC*LENGTH
                             equ
           NUMBER $MSGS
                                     'BO',
                             equ
           OPR$ACTION$LOOPS
                                     1007
                             equ
           START$CODE
                                     '053h', /* ascii big S */
                             equ
           TEST$BYTE
                                     '85',
                             equ
           TIMING CONTROL
                                     1121;
        declare
           array1(ARRAY$SIZE) address,
           array2(ARRAY$SIZE) address,
           i address,
           insbuffer(BUFFER$SIZE) bute initial ('AAAAAAAAAAAAAAA');
            in*buffer*ptr address initial (0),
           out$buffer$ptr address initial (0),
           timing$looper address;
        declare
           mixedstupe structure (
              char$buffer(BUFFER$SIZE) byte,
              int$number address
     nmout: procedure (value, base, 1c, buf, width) external;
        declare
            (value, buf) address,
            (base, lc, width) byte;
        end nmout;
       * signed 16 bit divide (d2 /d1)
      divid: procedure (dl. d2) address external;
                                                                      B-51
         declare
            (d1, d2) address;
         end divid;
```

```
⇒ print character string until $ encountered
     or$buf: procedure (strp) external;
        declare
           strp address;
     and pr$bufi
      * print character at console
     urscon: procedure (ch) external;
        declare
           ch byte;
        end wr$con:
10
     /*
      print address variable at console with one leading space
      :7
     putdec: procedure (num);
        declare
0
           num address,
           buf(7) bute;
                                      /* scratch buffer for conversion */
        call nmout(num, 10, ', .buf, 6);
        buf(6) = '$';
        call pr$buf(.buf);
0
        end putdec:
      ≠ get the next byte from in$buffer, increment in$buffer$ptr
      */
     jet$byte: procedure byte:
        declare
           c byte;
        c = in$buffer(in$buffer$ptr);
        insbuffersptr = insbuffersptr + 1;
        if insbuffersptr > last(insbuffer) then
           in$buffer$ptr = 0;
        return c;
        end get$byte;
.0
      put character into buffer, increment out$buffer$ptr
     putsbyte: procedure (inschar, pb);
        declare
0
           inschar byte,
           (pb, pc) address,
           put$char based pc byte;
        pc = pb + outsbuffersptr;
                                    /* compute ptr to next cell in buffer
        putschar = inschar;
0
        outsbuffersptr = outsbuffersptr + 1;
        if outsbuffersptr > BUFFERSSIZE - I then
           out#buffer#ptr = O;
        end put$byte;
        kernel is the main procedure in the benchmark program.
```

```
(temp1, temp2) address,
           i address,
           loop≱counter address,
           new$char byte,
           opr$action address,
           outsbuffer(BUFFERSSIZE) byte,
           while$counter address;
        io loop*counter = 0 to IO$LUCPS;
           output(IO$PCRT) = not TEST$BYTE;
        do loop*counter = O to last(array1);
           array1(loop*counter) = array2(ARRAY$SIZE-loop*counter);
           end:
        do loop$counter = 0 to NUMBER$MSGS;
           do while ((new$char := get$byte) <> START$CODE);
           while$counter = 0;
           do while (while*counter < MSG*LENGTH);
              if shr((new$char := get$byte), 1) then /* tests only bit 0 */
0
                 call put$byte(new*char, .out$buffer);
              else
                 call put$byte(new$char, .out$buffer);
              while$counter = while$counter + 1;
              end; /* do while */
           end; /* do loop$counter */
0
        opr#action = 0;
        do loop$counter = 0 to OPR$ACTION$LOOPS;
           if opr$action = 0 or opr$action = 3 or opr$action = 7 then
              mixed$type.char$buffer(opr$action) =
                 shr(TEST$BYTE, 3);
           else if opraction = 1 or opraction = 4 or opraction = 7 then
              mixed$type.char$buffer(opr$action) =
                 rol(TEST$BYTE. 2) and BITMASK;
           else if oprfaction = 2 or oprfaction = 6 then
              do i = 0 to last(mixed$type.char$buffer);
                 mixedstype.charsbuffer(i) = insbuffer(i);
                 end;
           else do;
                     /* default */
              temp1 = divid(INT2, INT1) * INT1;
              temp2 = divid(IN12, temp1);
              mixed$type.int$number = temp2 * INT2 + INT1;
              end;
           opreaction = opreaction + 1;
           if opr#action > 9 then
              opr#action = 0;
           end; /* do loop*counter * O to UPR$ACTION$LOOPS */
        end kernel;
     /******************
         start of main program
     <del>\########################</del>
        call pr$buf(.('Begin benchmark execution', CR, LF, '$'));
        do 1 = 0 to last(array2);
           array2(i) = i;
           end:
        do timing $100per = 0 to TIMING $CONTROL;
           call kernel;
           end;
        do i = 0 to BUFFERSSIZE - 1;
           call wr$con(mixed$tupe.char$buffer(i));
```

		ما المحاصلة في المستقدات المستقديديات المستقديد المستقدات المستقد			
	call pr≠bu÷(.('End E) end bmark;	cacution (CR, LF.	, (\$'));		•
.					
		·			
		· · · · · · · · · · · · · · · · · · ·			
					
			•		
, - ,					
()					
		·	·		
		·		· · · · · · · · · · · · · · · · · · ·	
		<u> </u>			
		·			
0					
					B-54
_					
· 				· •	

Tab 13 PLZ Source

9

<u></u>

0()

```
THIS IS THE MCTSSA DOT BENCHMARK IN PLZ. CREATED AT ANTHEM ON 2486PAG. DATA GATHERED ON ZOCTAG. USED ZDS 1740 MDS. PLZ REVISION H.
C
        OC:BENCHMARK MODULE

CONSTANT

ARRAYSIZE := 125; IOLOGPS := 575;

TIMINGCONTROL := 12; IOPORT :=01;

INTEGER1 := 300; INTEGER2 := -150;

SUFFERMAX := 13; STARTCODE := '8';
                                                                                                OPRACTIONLOOPS := 100;
NUMBERMOGS := 200; TESTBYTE := 85;
BITMASK := 1; MEGLENGTH := 30
CONOUT := 2;
Э
             TYPE
             CHAR BYTE: ! CHAR IS OF TYPE BYTE : BUFFERTYPE | ARRAYCBUFFERMAX + 1: CHARI; CHARBUFFER + BUFFERTYPE;
         EXTERNAL.
•
             PUTSTRING PROCEDURE(UNIT CHAR, PTR *CHAR);
PUTCHARS PROCEDURE(UNIT CHAR, PTR *CHAR, LEN WORD);
PUTINTEGER PROCEDURE(UNIT CHAR, INT INTEGER);
OUTPUT PROCEDURE(FIRST; CHAR, SECOND; CHAR);
SHIFTR PROCEDURE(THIRD; CHAR, FOURTH; CHAR)
ROTATL PROCEDURE(FIFTH; CHAR);
RETURNS (RSLT1; CHAR);
RETURNS (RSLT2; CHAR);
DIVID PROCEDURE(DIVISOR; INTEGER, DIVIDEND; INTEGER)
RETURNS(QUOTIENT; INTEGER);
             SEGIN GLOBAL (TO THIS MODULE) DATA DECLARATIONS !
         INTERNAL
             TIMINGLOOPER, LOOPCOUNTER, WHILECOUNTER, OPRACTION, INBUFFERPTR, OUTSUFFERPTR, TEMP1 , TEMP2 : INTEGER;
             ARRAY1, ARRAY2 : ARRAYEARRAYSIZE + 1: INTEGER);
 O / ' INBUFFER : SUFFERTYPE;
             NEWCHAR : CHAR;
             MIXEDTYPE : RECORDE
                  CHARBUFFER : BUFFERTYPE;
INTNUMBER : INTEGER; ]
                                                                         ! END RECORD MIXEDTYPE !
             END DATA DECLARATIONS ! BEGIN INTERNAL PROCEDURE DECLARATIONS/DEFINITIONS !
        GETBYTE PROCEDURE RETURNS(RETURNBYTE : CHAR);
RETURNBYTE := INBUFFERCINBUFFERPTR1;
             INBUFFERPTR += 1;
IF INBUFFERPTR > BUFFERMAX THEN INBUFFERPTR := 0 FI;
ENO GETSYTE;
         PUBBYTE PROCEDURE (INCHAR: CHAR, CHARPTR: CHARBUFPTR); ! NOTE THAT CHARPTR POINTS TO TYPE BUFFERTYPE. IT SHOULD BE PASSED ! THE ADDRESS OF THE BEGINNING OF THE ARRAY UPON WHICH IT WILL OPERATE.
             ENTRY
CHARPTR+[OUTBUFFERPTR] := INCHAR;
OUTBUFFERPTR += 1;
IF OUTBUFFERPTR > BUFFERMAX THEN OUTBUFFERPTR := 0 FI;
ENO PUTBYTE;
         KERNEL PROCEDURE;
                  CAL
I: INTEGER; ! LOCAL COUNTER !
OUTBUFFER; BUFFERTYPE;
BOTH I AND OUTBUFFER ARE AUTOMATIC/DYNAMICALLY ALLOCATED !
                                                                                                                                                                 B-56
```

 $\ddot{}$

```
ENTRY
           COOPCOUNTER := 0;
                      IF (JOPCOUNTER > IGLOOPS THEN EXIT FI; OUTPUT(NOT(YESTSYTE), IOPORT); (JOPCOUNTER +=1;
           90
:)
           LOOPCOUNTER := D;
           90
                       IF LOOPCJUNTER > ARRAYSIZE THEN EXIT FI;
ARRAY1CLOOPCOUNTER1 := ARRAY2CARRAYSIZE - LOOPCOUNTER1;
LOOPCOUNTER += 3;
           05
           LOOPCOUNTER := 0;
           00
                       IF LOOPCOUNTER > NUMBERMSGS THEN EXIT #I;
                                       NEWCHAR := GETBYTE;
IF NEWCHAR = STARTCODE THEN EXIT FI;
                       WHALECOUNTER := 0;
                       00
                                       IF WHILECOUNTER >= MSGLENGTH THEN EXIT F1;
NEWCHAR := GETSYTE;
                                      IF (SHIFTR(NEWCHAR, 1) AND BITM
    PUTSYTE(NEWCHAR, #OUTBUFFER);
ELSE
    PUTBYTE(NEWCHAR, #OUTBUFFER);
                                                                             1) AND BITMASK) = 1 THEN
Э
                                       WHILECOUNTER += 1;
                               I END OF BLZ FORM OF DO-WHILE LOOP :
CC
                       LOOPCOUNTER += 1;
END OF FOR LOOPCOUNTER FROM O TO NUMBERMSGS LOOP
           OFFRACTION := 0;
LOOPCOUNTER := 0;
           00
                       IF LOOPCOUNTER > OPRACTIONLOOPS THEN EXIT FI;
                       IF OPRACTION
                                                                          MIXEDTYPE.CHARBUFFER[OPRACTION] := SHIFTR(TESTBYTE, 3);
                                       CASE 0,3,9
                                                              THEN
                                                                          MIXEDTYPE.CHARBUFFER[OPRACTION]
ROTATL(TESTBYTE, 2) AND BITMASK;
                                      CASE 1,4,7
                                                              THEN
                                       CASE 2,6
                                                              THEN
                                                                                        IF I > BUFFERMAX THEN EXIT FI;
MIXEOTYPE.CHARBUFFER(I] :=
   INBUFFER(I];
I += 1;
                                                                                        := O;
                                                                                     00
                                       EL3E
       TEMP1 := DIVID(INTEGER2 , INTEGER1) * INTEGER1;
TEMP2 := DIVID(INTEGER2 , TEMP1);
HIXEDTYPE.INTNUMBER :=
TEMP2 * INTEGER2 + INTEGER1;
NOTE: ALTHOUGH NOT CLEAR FROM THE DOCUMENTATION, THE DIVIDE SHOWN BELOW WILL
NOT WORK SINCE PLZ HANDLES NEGATIVE NUMBERS INTERNALLY AS THOUGH THEY WERE
LARGE POSITIVE NUMBERS. FOR EXAMPLE, 4/(-2) IS EQUAL TO ZERO. ******

| MIXEDTYPE.INTNUMBER := (((INTEGER1/INTEGER2)
| * INTEGER1/INTEGER2) + INTEGER1; |
                                ! END CASE !
                       OPRACTION += 1;
LOOPCOUNTER +=1;
                                                                                                                                           B-57
```

r

```
IF OPRACTION > 9 THEN OPRACTION := 0 FI;
C
       on ! END FOR LOOPLOUNTER FROM O TO OPRACTIONLOOPS!
     END KERNEL;
    ENTRY
PUTSIRING (CONOUT: #'BEGIN BENCHMARK EXECUTION.MR. 1);
       LOOPCOUNTER := 0;
              IF LOOPCOUNTER > ARRAYSIZE THEN EXIT FI;
ARRAYZILOOPCOUNTER: LOOPCOUNTER;
LOOPCOUNTER += 1;
        00
ં ⊃
       INBUFFERPTR := 0;
               IF INBUFFERPTR > BUFFERMAX THEN EXIT FI; INBUFFERCINBUFFERPTR1 := 'A'; INBUFFERPTR += 1;
        00
        INBUFFER[10] := STARTCODE;
        INBUFFERPIR := 0;
OUTBUFFERPIR := 0;
        TIMINGLOOPER := 0;
               IF TIMINGLOOPER > TIMINGCONTROL THEN EXIT FI;
KERNEL;
 0
               TIMINGLOOPER += 1;
        00
       PUTCHARS(CONOUT, %MIXEDTYPE.CHARBUFFERLO), BUFFERMAX + 1);
PUTINTEGER(CONOUT, MIXEDTYPE.INTNUMSER);
       PUTSTRING (CONOUT, #'MREND EXECUTION.MR/');
     ENO MAIN!
     END DOTBENCHMARK; ! END MODULE !
```

0

J

Ċ

APPENDIX C DCT/HOL STUDY BENCHMARK PROGRAM RESULTS

This appendix contains information generated by running benchmark programs using the candidate languages.

o .

 \mathcal{L}

Table C-1. DCT HOL Summary Benchmark Results

Language	Executive Time (Min: Sec)	# Bytes Absolute Object Code(No I/O)	Program Support Environment	Compile Time
Interactive C	:58.5	1 286	PDP-11/70 UNIX	: 45
Whitesmith C	1:00	2538	CP/M	5:17
FORTRAN-80	4.03	3570	CP/M	1:47
RATFOR	3:43	3925	CP/M	3:01
Pascal/MT	1:36	3298	CP/M	:52
Pascal/Z	2:18	2304	CP/M	3:01
PLI-80	2:30	4514	CP/M	2:17
PLMX	:59	1759	CP/M	7:00
PLZ	2:48 ⁽¹⁾	2165	ZILOĞ	4:00

⁽¹⁾ Corrected for 2.5 MHZ Z80

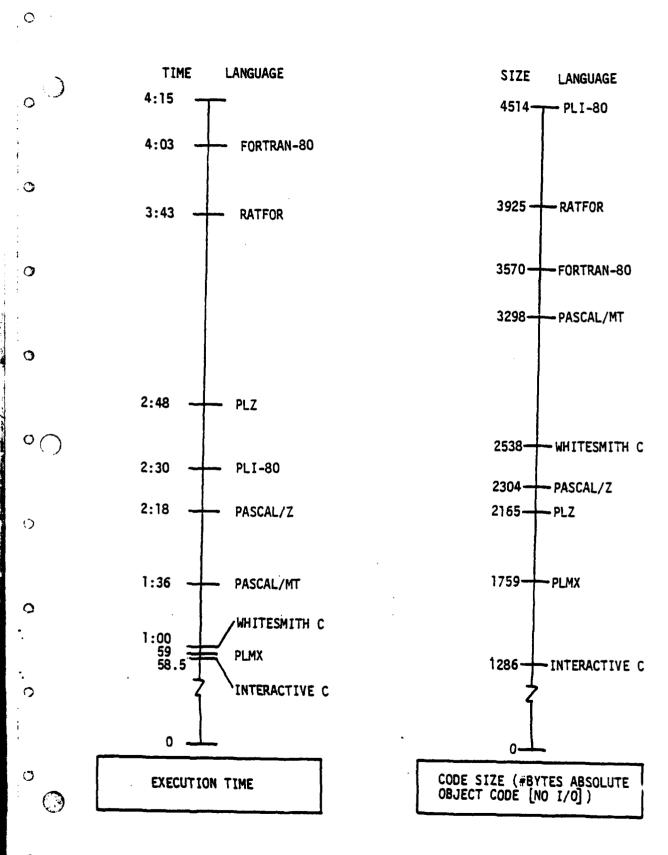


Figure C-1. Benchmark Results

APPENDIX D

DELPHI PHASE 1: DERIVATION OF WEIGHTS

This appendix presents the statistical results of Phase 1 of the Delphi study. The object of Phase 1 was to assign weights to the various language features.

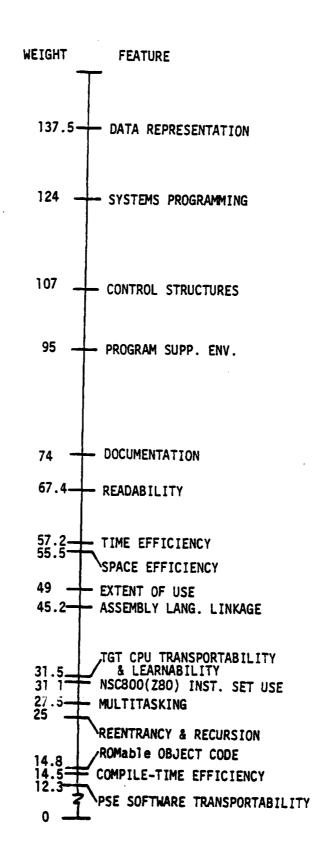
Table D-1. Phase 1 Delphi Study Statistical Summary

0

c 🔾

Ö.

	-	let Theretice							
					Zna iteration		2	MD Iteration	
LABOMAGE PEATURE	Average	Std. bev.	Std. Dav./ Average	Average	3td. Dev.	Std. Dav./	Average	Std. Dev.	Std. Dov.
DATA REPRESENTATION	121.3333	35.1034	2093	132.5556	31.0447	.2X2	137.5	43.6049	.3171
STOTES PROGRAMING	128.4444	41.6297	.3241	125.222	30.7481	.2455	124	29.5146	.2380
CONTROL STRUCTURES	115.0000	27.6134	.2401	110.6667	22.4221	.2026	101	21.2393	2661
PROGRAM SUPP. IMV.	108.6667	37.6132	.3461	101.7778	27.5172	.2704	8	16.4991	.1737
BOCUMENTATION	63.4444	28.2051	.4446	64.777B	22.1460	. M19	2	17.6069	2579
REA DASILITY	52.6667	20.8447	.3958	62.3333	20.3531	.3265	67.4	18.7451	.2781
TIME MFTICIBACT	43.4444	19.7934	.45%	50.6667	19.0919	. 3768	57.2	17.6814	2002
SPACE RPFICIENCY	40.2222	16.6717	.4145	51.6667	20.1556	.390t	55.5	16.9066	3046
EXTENT OF USE	51.0000	29.6564	.5815	47.4444	30.9561	.6525	65	30.0222	1219.
ASSEMBLY LANG. LINKAGE	50.3333	29.8831	.5937	47.7778	22.5875	.4728	45.2	13.2145	.2924
TABORT CRU TRAMSPORTABILITY	47.1111	22.6743	9269.	39.2222	20.1791	.5279	31.5	14.3469	.4556
LEARIADELITY	34.77/8	18.1437	.5217	35.5556	20.4641	.5755	31.5	13.6646	.4338
MSC 800 (260) (MST. SET USE	32.7T/B	20.4559	.6241	35.3335	20.2176	.5722	3.5	15.4944	4339
HULFITASKING	24.3333	16.2635	1899	22.0000	11.7047	.5320	27.5	26. 5744	1650
REMOTERANCE & PROCESSION	32.0000	12.5897	.3934	27.3333	8. 3217	3045	2	5.2705	2108
SOMeble Object Cops	17.000	5.8737	.3455	15.3333	9.3274	.6083	14.8	9.3106	.5615
COMPILE-TIME RPICIENCE	20,0000	9.4736	.47.57	16.8909	7.2015	.4264	÷.	3.1710	7912.
PSE SOFTWARE TRANSPORTABILITY	17.444	A. 945B	.512H	14.41	9.7225	1873.	12.3	+	92.5
							•	-	2



0 .

Figure D-1. Ranking of Features

Table D-2. First Iteration DCT NOL Language Peature Response Summary

်

°0

Particulate Peature Parture Pa								'	1		,		1	
Paris Pari	LANGUAGE PRATURE		RES	PONSI	ı	RDERI		ဥ္	ноп		TOTAL	AVERAGE	Std. dev.	ST JEV./ Average
15 95 90 100 150 150 150 150 150 151 1	SYSTEMS PROGRAMMING	86	8	ક્ર	110	120	130	136	190	200	1156	128.4444	41.6297	.3241
AGE 65 100 110 120 150 150 160 175 160 105 115 170 180 87 180	DATA REPRESENTATION	75	85	8	5	120	150	150	152	170	1092	121.3333	35.1034	.2893
70 80 86 95 100 150 160 170 180	CONTROL STRUCTURES	8	85	100	100	110	120	130	150	160	1035	115.0000	27.6134	.2401
AGR AGR AGR AGR AGR AGR AGR AGR		70	80	98	88	95	5	130	160	175	978	108.6667	37.6132	.3461
AGR 50 60 60 76 67 76 76 77	DOCUMENTATION	Š	35	46	22	8	65	70	105	110	172	63.4444	28.2051	.4446
AGR 20 1 10 35 50 63 70 75 85 459 459 51.0000 51.000	RBADABILITY	15	35	40	52	55	9	99	92	83	474	52.6667	20.8447	.3958
AGR 20 21 25 40 45 52 55 56 60 105 453 50.3353 5 TABILITY 5 20 29 45 60 65 70 110 424 47.11111 23 TABILITY 5 20 29 45 60 65 70 70 71 111 71 <td< th=""><th>8</th><th>-</th><th>10</th><th>35</th><th>Š</th><th>63</th><th>70</th><th>70</th><th>75</th><th>85</th><th>459</th><th>51.0000</th><th>29.6564</th><th>.5815</th></td<>	8	-	10	35	Š	63	70	70	75	85	459	51.0000	29.6564	.5815
TABILITY 5 20 20 45 60 65 70 110 424 47.1111 2 15 20 25 36 55 56 60 70 391 43.4444 1 15 26 30 30 40 47 50 60 65 36 40.2222 1 SSFT USB 15 15 35 35 35 50 60 65 313 34.7778 1 SSFT USB 6 15 35 36 35 35 35 35 35 37 36 37.7778 1 SSFT USB 6 15 30 36 35 35 35 35 36 37.7778 3 SSFT USB 6 15 30 36 35 48 35.0000 1 SSFT USB 10 16 15 15 16 20 20 20 36 <th>ASSEMBLY LANG. LINKAGE</th> <th>20</th> <th>21</th> <th>25</th> <th>40</th> <th>45</th> <th>52</th> <th>55</th> <th>8</th> <th>105</th> <th>453</th> <th>50.3333</th> <th>29.8831</th> <th>1265</th>	ASSEMBLY LANG. LINKAGE	20	21	25	40	45	52	55	8	105	453	50.3333	29.8831	1265
15 20 25 36 55 55 60 70 391 43.4444 1 1 1 1 2 2 2 3 3 3 4 4 4 4 5 6 6 5 6 3 5 3 4 0.2222 1 2 2 3 1 2 3 5 3 5 5 5 6 6 5 3 3 3 3 4.7778 1 2 2 3 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3	TARGET CPU TRANSPORTABILITY	2	82	ଷ	29	45	09	65	70	110	424	47.1111	32.6743	9269.
15 25 30 40 47 50 60 65 362 40.2222 1 SBT USB 15 15 35 35 35 50 60 65 313 34.7778 1 SBT USB 6 15 25 30 34 35 35 80 295 32.7778 1 ION 15 20 28 30 36 35 35 60 288 32.0000 1 ION 15 10 15 15 15 20 21 28 35 35 36 20 288 32.0000 1 BINCY 10 12 15 15 20 20 20 36 40 180 20.0000 1 SINCY 1 15 15 15 12 20 20 20 20 20 20 20 20 35 157 17.4444	TIME SPPICIBNCY	15	20	25	36	55	55	55	09	70	391	43.444	19.7934	.4556
SIBT USIÈ 6 15 15 36 35 36 50 60 65 313 34.7778 1 SIBT USIÈ 6 15 26 30 34 35 35 35 35 35 35 35 37 37.7778 2 ION 15 20 28 30 30 35 35 35 60 288 32.0000 1 BINCY 10 10 15 15 15 20 21 25 48 55 219 24.3333 1 BINCY 10 12 15 15 16 20 20 20 30 40 180 20.0000 1 9 10 15 15 15 20 20 25 25 153 17.0000 1	SPACE EPPICIBNCY	15	25	S.	8	40	47	50	8	65	362	40.2222	16.6717	.4145
SRT USB 6 15 25 30 34 35 35 35 36 295 32.7778 2 ION 15 20 28 30 30 35 35 60 288 32.0000 1 ION 10 10 15 15 20 21 25 48 55 219 24.3333 1 INCY 10 12 15 15 18 20 20 30 40 180 20.0000 9 ONTABILITY 2 10 15 15 20 20 20 35 157 17.4444 9 10 14 15 15 20 20 25 25 153 17.0000	LEARNABILITY		15	15	35	35	35	50	50	65	313	34.7778	18.1437	.5217
ION 15 20 28 30 35 35 35 35 60 288 32.0000 1 RINCY 10 10 15 15 20 21 25 48 55 219 24.3333 1 BINCY 10 12 15 15 18 20 20 30 40 180 20.0000 ONFABILITY 2 10 15 15 20 20 20 35 157 17.4444 9 10 14 15 15 20 20 25 25 153 17.0000	800 (Z80) INST. SET	9	15	22	8	34	35	35	35	8	295	32.7778	20.4559	.6241
BINCY 10 15 15 20 21 25 48 55 219 24.3333 1 BINCY 10 12 15 15 18 20 20 30 40 180 20.0000 ORFABILITY 2 10 15 15 15 20 20 20 35 157 17.4444 ORFABILITY 9 10 14 15 15 20 20 25 25 153 17.0000	REBUTRANCY & RECURSION	15	8	28	8	30	35	35	35	99	288	32.0000	12.5897	.3934
BINCY 10 12 15 15 18 20 20 30 40 180 20.0000 ORFABILITY 2 10 15 15 20 20 20 20 35 157 17.4444 9 10 14 15 15 20 20 25 25 153 17.0000	MULTITASKING	01	9	15	15	20	21	25	48	55	219	24.3333	16.2635	. 6684
ORTABILITY 2 10 15 15 20 20 20 20 35 157 17.4444 9 10 14 15 15 20 20 25 25 153 17.0000	COMPILE-TIME BPPICIBNCY	10	12		15	18	20	20	30	40	180	20.0000	9.4736	.4737
9 10 14 15 15 20 20 25 25 153 17.0000	PSE SOFTWARE TRANSPORTABILITY	2	2	15	15	20	20	20	20	35	157	17.4444	8.9458	.5128
	ROMable OBJECT CODE	6	0	14		15	20	20	25	25	153	17.0000	5.8737	.3455

Table D-3. Second Iteration DCT HOL Language Feature Response Summary

O

· Ō

0

O

O

LANGUAGE PEATURE		REST	RESPONSES	85	ORDERED	₹Oĭ	TO HIGH	₩ ₩		TOTAL	AVERAGE	STD. DEV.	STD. DEV. / Average
DATA REPRESENTATION.	100	105	110	120	120	120	168	170	180	1193	132.5556	31.0447	.2342
SYSTEMS PROGRAMMING	35	100	100	110	120	130	135	150	- 8	1127	125.222	30.7481	.2455
CONTROL STRUCTURES	80	986	100	100	105	120	125	130	150	966	110.6667	22.4221	. 2026
PROGRAM SUPP. ENV.	80	90	81	85	90	100	110	130	160	916	101.7778	27.5172	.2704
DOCUMENTATION	40	55	55	55	58	9	70	8	110	583	64.7778	22.1460	.3419
READABILITY	35	38	40	55	70	80	80	90	83	195	62.3333	20.3531	.3265
SPACE RPICIENCY	8	8	35	40	50	90	75	15	80	465	51.6667	20.1556	.3901
TIME SPPICIENCY	20	35	35	50	55	54	25	42	8	456	50.6667	19.0919	.3768
ASSEMBLY LANG. LINKAGE	21	30	31	40	43	45	09	65	8	430	47.7778	22.5875	.4728
EXTENT OF USE	-	5	30	40	50	63	92	11	95	427	47-4444	30.9561	.6525
TARGET CPU TRANSPORTABILITY	5	15	30	32	40	45	55	65	65	344	38.2222	20.1791	.5279
LEARMABILITY	13	20	20	22	30	35	45	69	70	320	35.5556	20.4641	.5755
NSC 800 (Z80) INST. SET USE	9	25	30	32	30	33	34	50	08	318	35.333	20.2176	.5722
REBUTRANCY & RECURSION	13	15	25	28	30	30	35	35	35	246	27.3333	8.3217	.3045
MULTITASKING	2	12	15	15	20	23	25	30	48	198	22.0000	11.7047	.5320
COMPLEA-TIME BPPICIENCY	9	2	10	12	17	18	20	25	30	152	16.8889	7.2015	.4264
ROMable OBJECT CODE	4	5	10	10	14	15	25	25	30	138	15.3333	9.3274	.6083
PSE SOFTWARE TRANSPORTABILITY	2	9	10	10	12	15	20	20	30	130	14.444	9.7225	.6731
	1	1	1		7]							

Table D-4. Third Iteration DCT HOL Language Peature Response Summary

LANGUAGE PRATURE		RESP	RESPONSES	- 0RI	ordered	i Mot	To High	Ħ			TOTAL	AVERACE	STD. DEV.	STD. DEV. / Average
DATA REPRESENTATION	2 0	105	110	110	120	120	125	170	180	235	1375	137.5	43.6049	1216.
SYSTEMS PROGRAMMING	85	100	110	110	115	115	130	135	150	190	1240	124	29.5146	.2380
CONTROL STRUCTURES	88	90	06	100	105	110	115	120	120	150	1070	107	21.2393	1985
PROGRAM SUPP. ENV.	75	08	98	85	8	90	100	100	120	125	950	96	16.4991	1221.
DOCUMENTATION	25	55	09	65	99	75	80	85	8	110	740	74	17.6068	.2379
RRADABILITY	35	45	50	90	Ô,	86	90	90	98	88	674	67.4	18.7451	.2781
TIME RPTICIBLICY	20	45	47	50	55	09	70	75	75	75	572	57.5	17.6874	. 3092
SPACE EPPICIBNOY	30	35	45	45	50	99	70	70	70	80	555	55.5	16.9066	.3046
BXTENT OF USE	1	5	9,0	40	50	09	64	70	80	8	490	64	30.0222	.6127
ASSEMBLY LANG. LINKAGE	21	31	40	04	45	45	50	55	09	65	452	45.2	13.2145	.2924
TARGET CPU TRANSPORTABILITY	5	15	25	ο <u>κ</u>	0%.	35	32	40	45	55	315	31.5	14.3469	.4556
LBARNABILITY	13	20	20	22	30	35	35	40	40	09	315	31.5	13.6646	.4338
NSC 80C (280) INST. SET USB	9	25	25	0%	30	30	30	30	95	55	311	31.1	13.4944	.4339
HULTITASKING	10	10	12	20	20	23	25	25	30	100	275	27.5	26.3744	1656.
RESPIRANCY & RECURSION	15	20	20	25	25	25	30	30	30	%	250	25	5.2705	.2108
ROMable Object cons	4	5	10	10	14	15	15	20	52	30	148	14.8	8.3106	.5615
COMPLIE-TIME RPPICIENCY	10	10	12	15	15	15	15	15	18	20	145	14.5	3.1710	.2187
PSE SOPTWARE TRANSPORTABILITY	2	9	10	10	10	10	10	15	20	30	123	12.3	7.8322	.6368

APPENDIX E

DELPHI PHASE 2: DERIVATIONS OF SCORES AND FIGURES OF MERIT

This appendix presents the statistical results of Phase 2 of the Delphi study. The object of Phase 2 was to assign gross scores and figures of merit to each language based upon the weights of the various language features assigned in Phase 1.

0

Ö

AD-A118 811 MARINE CORPS TACTICAL SYSTEMS SUPPORT ACTIVITY CAMP --ETC F/6 9/2
DIGITAL COMMUNICATIONS TERMINAL HIGH ORDER PROGRAMMING LANGUAGE--ETC(11)
NOV 80 K C SHUMATE, R E SAUER, G E ANDERSON
UNCLASSIFIED 24E005/U-TN-01-VOL-2
NL

Table E-1. Delphi Study -- Phase 2 Summary (1st Iteration)

()

Language Punture	耳	8436	ပ	COROL	PORT-66	LJLIM	MATPOR	PASCAL	POKTH	PL/1-80	PLHA	714
Buta Represents tion	2	33.6875	e9.3750	79.0625	60.885	68.75	42.5	114.4687	59.57BI	010	El.Ulel	01
System	24.B	15.934	110.36	27.28	47.5292	46.5	55.8	BO. 6000	94.2400	9.09	9.111	9.08
Control Structures	45.8	34.1009	M. 8381	53.5	41.2699	72.76	74.9	95.9375	96.3400	£.£	49.1631	. 6
Program Supp. thes.	2	72.0345	76.35	53.2	98.9	57	66.5	74.4175	47.5000	26	12.33	66.5
Posuments tion	29.6	54.1162	56.1142	61.42	52.6436	50.38¢	53.65	63.3662	30.2142	57.165	56.7284	62.9
Bundability	6.74	37.4678	46.0544	53.92	32.2576	51.744	¥.75	59.3929	19.0944	55.268	53.92	57.29
Time Efficionay	11.44	5.434	36.6080	17.16	33.8452	26.455	36.608	35.1322	33.1700	42.9	41.9448	40.0
Space Meliciency	25.2	15.0627	78.45	19.425	36.075	31.9125	40.2375	33.6941	44.8606	æ.93	38.85	33.3
Extent of Sue	6.4	44.1	28.5817	39.5	37.7946	33.32	30.625	35.5250	9.8000	29.4	37.5634	29.4
Assembly Language Liskage	4.%	10.6536	₹.52	17.176	34.352	78.7	72.71	25.6600	31.6400	72.71	\$.3	36.16
Terpet OPU Transport- ability	12.6	23.94	22.5729	16.5575	24.005	20,475;	20.475	15.5263	22.0500	17.325	25.2	16.7989
Luncushilty	3.15	26.5387	24.6739	21.001	20.0245	19.845	25.2	24.8062	12.6000	21.6562	25.2	26.775
MUC 800 (201) Inst. Set Nov	12.44	6.997	14.5112	6.22	15.55	15.95	15.55	16.9456	10.3656	11.6625	21.77	15.55
Multitanting .	2.75	13.75	9.6250	5.5	6.1875	B.25	8.25	7.5625	315.5415	9.1657	9.1657	13.75
Mountransy & Securation	•	3.3325	21.25	9	2.75	1.6675	1.6675	10.5000	16.2500	16.25	15	13.75
BUMmble Object Code	•	6.216	1:1	×.	10.064	35 35	9.25	10.0640	B. 3250	3.33	1.0%4	6.00
Compile-time Officiency	4.3	3.335	11.2575	3.8657	9.4250	8.5181	6.9421	B. 9900	9.5700	1.0667	7.25	7.9rs
fil Softante Tramport- ability	2.46	9.7908	3.4	7.78%	10.47%	8.9175	6.61	10.3320	7.9950	9.225	6.765	×.
TUTALS	277. T50U	416.5157	728.3219	488.1773	534.9217	534.9217 559.6992 610.2901 663.5567 539.1750 708.5161	610.2901	663.5567	539.1750	708.5161	726.4505	651.2907
¥												

Table E-2. Delphi Study--Phase 2 Summary (2nd Iteration)

O

7

 \bigcirc

0()

·.

Ó

Ó

Language Posture	Inter- active C	Mite-	Hierosoft FORT-60	Zilog FORT	UCSD FORT	Cromenco RATFOR	Pascal/HT Pascal/2	Pascal/2	PL/1-80	PLAG	27.4
Note Represes- tation	97.2262	92.8125	62.9062	52.25	59.125	68.73	118.8412	104.8437	110.55	69.4375	8.25
System Pro- Greening	112.6292	111.6	74.09	71.5	68.2	72.22	110.5708	79.3600	76.88	10.05	71.5
Coetrol Structures	90.0512	84.7119	53.5	51.36	57.78	75.6597	93.6497	86.0931	75.7881	86.9375	9.6
Program Supp. May.	71.9625	69.35	68.875	42.75	54.625	69.5115	22	74.1	76.95	68.875	52.25
Documentation	ळ.०६	56.24	54.76	46.25	51.8	51.8	59.5	55.5	53.65	51.8	51.8
Readel 11ty	48.1438	47.18	36.2275	39.082	41.788	42.3676	59.2176	57.29	51.6756	51.6756 49.7075	48.528
Extent of Use	26.46	25.725	36.75	18.0075	18.375	22.05	27.6017	22.05	23.52	40.0183	26.1317
-Just Liamesh	45.94	38.42	36.16	21.47	21.47	25.312	35.256	25.312	72.77	43.6903	41.4303
Parget CPU Frameport- ability	18.1125	23.625	25.2	22.05	12.6	21.2625	22.68	20.79	24.1511 31.5	¥.5	25.2
Learnebility	24.7496	25.2	24.3330	21.2625	22.68	22.5004	25.6504	24.9386	20.4750	23.625	22.8375
MSC 600 (280) Inst.Set Nee	26.4350	23.325	27.99	29.545	٥	•	26.435	27.99	24.86	13.995	31.1
Bulti tesking	0	. 6875	.9157	.9157	9	e	.6875	.9156	.9157	٥	•
Boutrascy & Recursion	21.6675	18.875	.8325	1.25	0	•	20.625	23.3325	16. 5325	2	æ
Rollable Object Code	0	0	13.32	۰	•	0	6. 928B	6.9286	7.955	14.8	14.8
PME Software fremaport- ability	6.9704	9.84	9.6	1.845	5.535	10.1475	2.5	9.4304	9.	6.9704	1.85
TOPALS*	637.6679	627.5919	\$25.7007	419.XT7	413.978	481.6906	481.6906 695.3839	552.8749		608.333 636.4065	600.2725

These totals do not reflect the inclusion of apace efficiency, time efficiency, or compile-time efficiency.

Table E-3. Delphi Study--Phase 2 Summary (3rd Iteration--MCTSSA)

.

. (3

Ģ

)

Ö

O

5)

bits blaces 115.689 110 62.6912 94.29 123.79 129.659 111.9529 110.2924 110.2	Language Postturo	Inter- Active C	unite-	1101080ft 1089-80	804178 Crossoo	18/1000A	2/12044	08-1/74	xxna	214
109.539 110.5564 51.6564 64.0564 103.7364 94.0292 103.5292 10	Mais Repressa- tation	115.885	110	62.0512	\$6.25	123.75	129.635	111.9525	79.5457	91.3275
## 55.783 78.037 81.764 77.6501 38.1277 87.1194 87.1194 ### 62.9 54.4764 57.72 59.94 61.42 62.9 54.02 51.8 ### 62.9 54.4764 57.72 59.94 61.42 62.9 54.02 51.8 ### 57.72 54.4764 57.72 59.94 61.42 62.9 54.02 51.8 ### 57.72 54.4764 57.72 59.94 61.42 62.9 54.02 51.8 ### 57.72 54.4774 13.764 15.036 34.6376 24.2471 22.308 56.7139 ### 55.5 28.128 19.9911 18.1818 21.639 30.9745 15.813 40.576 ### 28.5817 35.4824 24.2991 28.75 37.1273 34.5418 34.8672 ### 24.7825 24.582 0 0 0 0 0 0 0 0 0 0 0 ### 24.5825 24.582 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	System Pro-	109.529	110.5584	31.6584	1950-19	108.6984	103.7384	#.02%	103. 3292	96.1708
as 63.785 78.035 78.035 78.035 78.035 78.035 78.035 79.922 79.922 79.922 78.705 as 63.9 94.0764 97.72 59.94 61.42 62.9 54.02 51.8 as 93.92 54.877 42.369 49.5861 60.66 60.7543 95.408 51.8 as 55.5 28.127 13.766 15.0036 34.6776 24.2471 22.308 96.7136 as 26.5 28.201 35.0091 16.1818 21.6394 30.9745 15.6119 40.576 as 28.201 35.1622 27.4409 30.3472 37.1273 34.3417 26.1396 as 28.201 35.202 27.409 30.3472 37.1273 34.2409 24.7495 as 36.202 27.409 19.9409 21.105 11.6125 18.9 24.7495 as 36.202 27.409 27.409 27.409 27.5067 27.5067 27.7	Control Structures	\$6.3	8.3	1806.84	1694	1059 . 17	76.1297	87.11%	87.1194	36.3597
62.9 54.6794 57.72 59.94 61.42 62.9 54.02 51.8 57.2 55.77 13.769 15.0036 34.6776 24.2471 22.308 56.7139 57.2 56.72 13.769 15.0036 34.6276 24.2471 22.308 56.7139 57.2 56.72 13.469 15.0036 34.6276 34.6271 22.308 56.7139 58.5317 33.4894 41.2384 38.3817 33.6894 36.3817 35.4817 26.1317 58. 58.72 25.7009 35.1862 27.4409 30.3472 37.1273 34.3817 26.1317 58. 58.72 25.7009 35.1862 27.4409 21.105 11.8125 18.9 25.7229 58. 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	Program Support	63.783	78.033	78.033	76.033	78.053	79.952	79.952	78.7075	49.533
53.92 54.877 42.366 49.5861 60.66 60.7343 55.8476 55.4567 57.2 55.77 13.768 15.0036 34.6576 24.2471 22.308 56.7136 56.5 28.128 19.9911 18.1818 21.6394 30.9745 15.6119 40.576 40 28.5417 33.1828 27.4409 30.3472 37.1273 34.4817 26.1317 41 38.0345 29.7009 35.1828 27.4409 30.3472 37.1273 34.5418 34.5812 5.2478 12.6 23.0389 19.9489 21.105 11.8125 18.9 24.7495 5.2478 24.2391 28.35 28.35 21.5995 24.7495 5.2478 3.5262 0	Becimentation	6.2.9	54.8784	51.72	59.94	61.42	62.9	54.02	51.8	58.9484
57.2 55.77 13.76a 15.00% 34.6576 24.2471 22.308 56.7136 40.55.5 28.128 19.9911 18.1818 21.6394 30.9745 15.6119 40.576 40.55.5 28.5417 33.4817 33.4817 35.4817 26.1317 26.1317 40 28.5417 27.089 19.949 21.105 11.8125 18.9 23.7259 41 26.5512 3.6262 3.6362 3.6362 3.6362 3.7.1273 34.5418 34.6872 40 26.5512 3.6262 3.6362 21.105 11.6125 18.9 23.7259 40 0	Readability	53.82	54.877	42.3609	1985.64	99.09	60.7543	95.8476	55.4567	56.8114
25.5 28.128 19.9911 18.1816 21.6394 30.9745 15.8119 40.576 4-244 28.2917 35.0884 30.9745 15.8119 40.576 40.576 4-244 24.284 36.2017 37.4817 37.4817 37.4817 26.1317 4-3 36.945 29.7009 35.1822 27.4409 21.105 11.8125 18.9 27.7229 3-4.7495 25.4236 24.3432 24.2499 21.105 11.8125 18.9 27.7229 4-3 24.7495 24.2496 21.105 11.8125 18.9 27.729 5-4-7495 24.3432 24.2496 21.105 11.8125 18.9 27.729 6-4-7495 3-6476 3-6476 3-6476 3-6476 3-6476 3-7.29 3-7.729 1-4-7405 3-6476 3-6477 3-6476 3-6477 3-6476 3-6476 3-7.645 3-7.645 3-7.645 3-7.645 3-7.645 3-7.645 3-7.645 3-7.645 3-7.645	Time Efficiency	57.2	75.77	13.768	15.0036	34.0576	24.2471	22.308	56.7130	19.917
ab. Objets 280-5817 35-3884 36-3817 35-3884 36-3817 35-4817 35-4817 35-4817 35-4818 41-2784 36-3817 37-1273 34-5418 26-1317 111y 5-2478 12-6 25-0989 19-9489 21-105 11-8125 18-9 25-7229 111y 5-2478 12-6 25-0989 19-9489 21-105 11-8125 18-9 25-7229 10-6 25-2512 24-3432 24-2991 28-35 28-35 21-5995 24-7495 10-6 0	Space	55.5	28.128	1166.61	18.1818	21.6394	30.9745	15.6119	40.576	32.967
36.0945 29.7009 35.1862 27.4409 30.3472 37.1273 34.5418 34.8672 34.0945 25.0969 19.9489 21.105 11.8125 18.9 25.7229 24.7495 24.3432 24.2991 28.35 28.35 21.5995 24.7495 24.7495 26.3512 26.352 21.5995 24.7495 24.3432 24.3432 24.2991 28.352 23.0637 3.6262 27.399 24.3625 0 0 0 0 0 0 0 0 0	Extent of the	28.5417	33.dBe	41.2384	34.3617	33.8884	78.3817	33.4817	26.1317	35.525
11ty 5.2478 12.6 23.0999 19.9489 21.105 11.8125 18.9 25.7229 24.7495 25.4236 24.3432 24.2991 28.35 28.35 21.5995 21.7995 24.7495 34.5826 3.6262 0 1	Assembly Las- guage Linkage	38.0945	29.7009	35.1082	27.4409	30.3472	37.1273	34.5418	74.8672	¥6.16
24-7495 26-755 24-3432 24-2991 28-35 28-35 21-5935 21-5935 24-7495 24-7495 0 0 0 0 25-3962 23-0637 3-6262 27-39 0 0 0 0 0 0 0 0 0 st 24-5825 0<	Target CFU Transportability	5.2478	12.6	23.0989	19.9469	21.105	11.8125	18.9	25.7229	6.615
1.00 1.00	Learnability	24.7495	25.4236	24.3432	24.2991	28.35	28.35	21.5995	24.7495	24.9732
of 0	MSC 300 (280) [Ast. 3et Nee	26.9512	3.6262	3.6262	•	25.3962	23.0637	3.6262	27.39	27.99
24.5825 20 0 22.5 20.415 2.5 0 14 7.48216 14.5524 17.926 5.92 14.5528 13.0728 7.6456 13.32 14.5 2.0575 6.0967 3.6047 12.5463 3.6047 4.7532 1.5529 144.5 5.3895 9.84 11.172 11.172 11.2741 11.2742 11.2741 10.455 663.675 659.875 744.8101 527.959 955.7995 717.8052 718.6067 616.4996 619.1928 790.8755 744.8101 527.9599 993.5695 786.8905 777.433 659.3727 718.0355	Sulttenting	0	0	0	0	•	o	•	۰	0
14-5 2-097 3-604 12-5528 13-0728 7-6456 13-32 14-5 2-0575 6-0967 3-6047 12-5483 3-6047 4-7632 1-5529 144-5 2-0575 6-0967 3-6047 12-5483 3-6047 4-7632 1-5529 144-5 3-549 11-172 11-174 11-2741 11-2741 10-455 663-675 659-875 440-029 556-7995 717-8052 718-5067 616-4096 619-1928 790-875 744-8101 527-9595 593-5996 786-8055 777-433 659-3727 718-0355	Reentracy 4 Securator	24.5825	24.5825	0	0	22.5	20.415	2.5	0	æ
14.5 2.0575 6.0987 3.6047 12.5483 3.6047 4.7632 1.5529 114y 5.3895 9.84 11.172 11.2741 11.2742 11.2741 10.455 663.675 658.850 480.020 556.7995 717.8052 718.6067 616.4096 619.1928 790.8755 744.8101 527.9595 593.5096 786.8095 777.433 659.3727 718.0355	Molecule Object Code	7.4216	14.5524	7.928	5.8	14.5528	13.0728	7.6456	13.32	14.5528
Magnetability 5.3299 9.84 11.172 11.172 11.2741 11.2742 11.2741 10.455 00445** ORALS* 663.6799 656.3608 488.0929 556.7995 717.8052 718.6067 616.4096 619.1928 01 790.8795 744.6101 527.9599 593.5096 786.8095 777.433 659.3727 718.0355	Compile-time Efficiency	14.5	2.0575	6.0987	3.6047	12.5483	3.6047	4.7632	1.5529	2.7187
113* 663.6755 659.3608 448.0929 556.7935 717.8052 718.6067 616.4096 619.1928 790.8755 744.8101 527.9595 593.5096 786.8905 777.433 659.3721 718.0355	756 doftware Transportability	5.329	9.84	11.172	11.172	11.2741	11.2742	11.2741	10.455	5.32%
790.8755 744.8101 527.9595 593.5896 786.8505 777.433 659.3721 718.0355	10tal.3*	663.675	658.3608	456.0929	556.7995	717.8052	718.6067	616.4896	619.1928	617.2963
	POI	790.8755	744.8101	527.9595	9695 - 5696	106.0505	TT.433	659.3721	718.0355	672.399

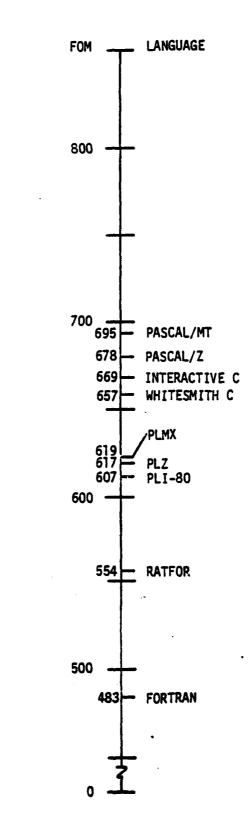
Table E-4. Delphi Study--Phase 2 Summary (3rd Iteration -- NOSC Included)

.5

Ċ.

France Latin 115.0375 111.62 65.6775 55.3877 110.62 65.6775 55.3877 110.62 65.6775 55.3877 110.62 65.6775 55.3877 110.62 65.6775 55.3877 110.62 65.6775 55.08 66.2 65.6775 110.62 65.6775 55.0878 110.62 65.6775 110.62 65.6775 110.62 65.6775 110.62 65.6775 110.62 65.6775 110.62 65.6775 110.62 65.6775 110.62 65.6775 110.62 65.6775 110.62 65.6775 110.62 65.6775 110.62 65.6775 110.62 65.0775 110.62 65.6775 110.62 65.0775 110.62 65.0775 110.62 65.0775 110.62 65.0775 110.62 65.0775 110.6775 1		ster-	*.437m	Reroeft	9	Passal/RR	7/Isseef	;		
### 115.0875 111.82 65.6779 95.3837 Reares 110.9056 112.382 95.0488 68.285 ### 15.0495 82.8891 857.885 82.885 ### 15.0495 82.8851 857.885 82.8851 857.885 ### 15.0495 82.8851 857.885 ### 15.0495 827.8851 857.885 ### 15.0495 827.8851 857.8852 482.8891 857.8855 827.8851 857.8855 827.8851 857.8855 827.8851 857.8855 827.8851 857.8855 827.8851 857.8855 827.8851 857.8855 827.8851 857.8855 827.8851 857.8855 827.8951 857.8851 857.8855 827.8851 857.8855 827.8851 857.8851 857.8855 827.8855 827.8851 857.8855 827.8851 857.8855 827.8851 857.8855 827.885		ני	.					7.1-d0	PLAS.	214
Attached 110.9056 112.282 55.0168 662.955 But and a 55.979 93.9246 49.9569 82.955 But and a 60.6428 55.073 56.552 56.5696 Blitty 52.972 53.1719 36.332 56.5699 Blitty 52.972 53.1719 36.332 15.0055 Bloomy 57.2 53.77 13.766 15.0075 But and 26.3122 34.6036 35.2568 35.7014 But and 25.972 32.05 19.3504 But and 25.972 12.7953 22.05 19.3504 But and 1.375 1.375 0 0 But Code 9.62 14.1651 6.7829 5.0734 Ctuare 14.578 5.3013 9.455 11.1573 But and 1.378 5.3013 9.455 11.1573 But and 1.378 5.3013 9.455 11.1573 But and 1.378 5.3013 9.455 11.1573	procentation	115.0875	29.111	6.975	95. M37	123.75	5212-151	110.0663	78.5437	31.3275
57. 97. 93. 9246 49. 9569 82. 855 60. 6428 55. 073 54. 546 75. 646 75. 6015 60. 6428 55. 073 54. 5562 56. 5956 57. 2 53. 77 13. 766 15. 0025 57. 2 53. 77 13. 766 15. 0025 57. 5 26. 1219 19. 99 16. 1816 26. 312 34. 6036 35. 205 19. 350 1. 375 12. 755 22. 05 19. 350 1. 375 1. 375 0 0 0 24. 1675 24. 0625 .3425 0 24. 1675 24. 0625 .3425 0 14. 5 2. 0575 6. 0387 3. 6047 14. 5 2. 0575 6. 0387 3. 6047 669. 89718 657. 2052 482. 5891 554. 3706	Series in Contract	110.90%	112.282	5.08	?	103.9988	8.8	\$2.11.56	103.3292	98.1708
60.6428 59.073 54.568 75.608 75.608 60.6428 59.073 54.558 75.468 75.608 59.289 59.073 54.558 75.668 75.608 59.289 77.2 59.1719 30.389 18.1818 20.3122 34.609 39.2898 39.7014 29.628 12.7953 22.05 19.7504 23.34 24.639 23.1356 24.214 24.1375 0 0 0 14.375 0 0 0 0 14.5 24.1675 24.0625 .3425 0 0 0 14.5 24.1675 6.0387 3.6047 14.5 2.0575 6.0387 3.6047 3.604 257.205 482.5891 554.3706	rectares	3.93	93.88	19.9369	22.25	\$5.6299	24.7592	35.9375	87.11	86.3591
57.2 55.77 13.764 15.0079 57.2 55.77 13.764 15.0079 57.2 55.77 13.764 15.0079 57.5 28.1218 19.39 18.1819 28.3122 34.6036 35.208 35.7014 28.6653 12.7953 22.05 19.3504 27.2654 4.0803 4.0803 0 6.24.1675 24.0525 .3925 0 6.24.1675 24.0525 .3925 0 6.09718 57.2054 482.5891 554.3706	ra gport freezent	67.7635	7.8%	ķ	75.404.5	iç \$	بر چ	76.6745	<u> </u>	49.533
97.2 55.77 13.766 15.0039 97.2 55.77 13.766 15.0039 95.5 28.1218 19.39 18.1818 28.3122 34.6036 35.9069 35.003 8.6635 12.7935 22.05 19.3504 27.2654 4.0803 4.0803 0 1.375 1.375 0 0 0 0 1.375 1.375 0 0 0 0 1.375 1.375 0 0 0 0 1.375 1.375 0 0 0 0 1.375 1.375 0 0 0 0 1.375 1.375 0 0 0 0 1.375 1.375 0 0 0 0 1.375 1.375 0 0 0 0 1.375 1.375 0 0 0 0 0 1.375 1.375 0 0 0 0 0 1.375 1.375 0 0 0 0 0 1.375 1.375 0 0 0 0 0 0 1.375 1.375 0 0 0 0 0 0 0 1.375 1.375 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	sectation.	60.0428	\$5.033	\$.7562	56.5856	57.6164	56.0912	56.1142	51.6	58.2484
95.5 28.1218 19.39 16.1818 20.3122 34.6030 39.2980 35.7014 30.625 12.7953 22.05 19.3504 23.34 24.633 23.135 24.214 1.375 1.375 0 0 0 1.375 1.375 0 0 0 1.4.67 24.0625 .9325 0 24.1675 24.0625 .9325 0 14.5 2.0575 6.0387 3.6047 14.5 2.0575 6.0387 3.6047 14.5 2.0575 6.0387 3.6047	ob1118y	52.572	53.1719	****	19.2829	59.5344	59.6086	53.52	55.4567	56.3114
26.3122 34.6036 59.2568 35.7014 26.5122 34.6036 39.2568 35.7014 26.663 12.7953 22.05 19.7504 27.264 4.0803 4.0803 0 24.1675 24.0625 .3925 0 24.1675 24.0625 .3925 0 14.5 2.0575 6.0387 3.6047 14.5 2.0575 6.0387 3.6047 669.8978 657.2052 482.5891 554.3706	fictonsy	57.2	55.TI	13.76	15.0035	x.677	24.2471	11.61	56.7152	19.917
8.625 12.7953 22.05 19.7504 25.296 35.7014 25.625 12.7953 22.05 19.7504 25.34 24.6399 25.1356 24.214 27.35 1.375 0 0 0 1.375 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	fletency	8.8	26.1216		16.1818	21.4174	30.9745	15.8119	40.576	15.967
98.625 12.7953 22.05 19.7504 25.94 25.94 25.94 25.94 24.6399 25.1375 24.214 1.375 1.375 0 0 0 1.375 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	at of the	28.3122	X4.6036	39.2388	35.7014	X.X	33.1975	29.4	26.1317	35.525
8.6629 12.7953 22.05 19.7504 23.34 24.6393 23.1336 24.214 1.375 4.0803 4.0803 0 1.375 1.375 0 24.1675 24.0625 .3425 0 9.62 14.1651 6.7888 5.0734 14.5 2.0575 6.0387 3.6047 14.5 2.0575 6.0387 3.6047 669.8978 657.2052 482.5891 554.7706	ably agenge akage	38.8268	X0.906e		29.0952	30.6366	X0. X	75.3148	X.8572	X6.16
23.34 24.6399 23.1336 24.214 1.375 1.375 0 0 24.1675 24.0625 .3325 0 24.1675 24.0625 .3325 0 14.5 2.0575 6.0387 3.6047 14.5 2.0575 6.0387 3.6047 669.8978 657.2052 482.5891 554.3706	pet CF6 masporta- ilaty	8.6625	12.7953	L	19.3504	21.1491	3.58	18.4496	25.7229	6.615
1.375 1.0803 1.0803 0 0 0 0 0 0 0 0 0	mability	23.34	24.6393	L	24.214	27.1246	27.1247	21.2625	<u> </u>	
1.375 1.375 0 0 0 24.1675 24.0625 .3425 0 9.62 14.1651 6.7888 5.0734 14.5 2.0575 6.387 3.6047 4.578 9.3013 9.495 11.1573 669.89718 657.2052 482.5891 554.3706	300 (330) st. 3et liee	27.2654	4.0803	4.0803	9	25.7137	19.2447	3.110	27.39	27.99
9.62 14.1651 6.7888 5.0734 14.5 2.0575 6.0387 3.6047 14.5 9.455 11.1573 669.8978 657.2052 482.5891 554.3706	l tauking	1.375	1.375	0	0	0	0	0	c	0
9.62 14.1651 6.7888 5.0734 14.5 2.0575 6.0987 3.6047 81114y 4.578 9.495 11.1573 669.8978 657.2052 482.5891 554.3706	trancy Recurston	24.1675	24.0625	. 3425	0	20.5%	19.22	5.000	•	ĸ
14.5 2.0575 6.0387 3.6047 billity 4.578 9.3013 9.455 11.1573 669.8978 657.2052 482.5891 554.3706	ble jest Code	9.62	14.1651	6. 734H	5.0734	13.505	12.6051	7.6472	13.52	14.5528
Software 4.578 9.3013 9.455 11.1573 ransportability 4.578 9.302 9.455 11.1573	tlotensy	14.5	2.0575	6.0987	3.6047	12.5483	3.6047	4.6732	1.5529	18:11.5
AL* 669.8978 657.2052 482.5891 554.3706	Software emaportability	4.578	9.3013	9.455	11.1573	9.6555	9.61	9.6641	10.455	6. 3295
	•7	669.9978	657.2052	182.5891	554.3706	694.653	678.3129	607.4803	619.1928	617.2965
743.1545 522.4458 591.1600		797.097B		522.445B	591.1606	765.4764	787.1812		647.3724 118.0369	612.333

"form! without space, time, compilertime efficiency.



. . . .

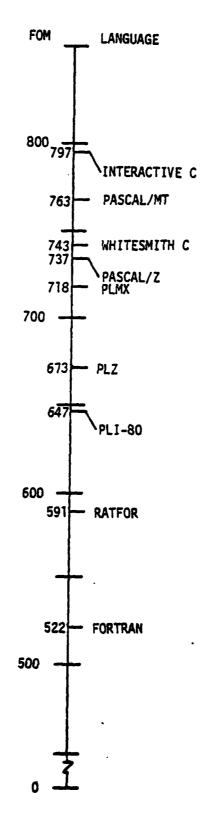
O

(3)

Ç

()

Figure E-1. Figures of Merit without Benchmark Values
E-6



, O

Ö

0

Mark E.

Figure E-2. Final Figures of Merit E-7

